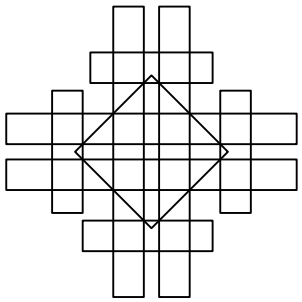
37th International Symposium on Computational Geometry

SoCG 2021, June 7–11, 2021, Buffalo, NY, USA
(Virtual Conference)

Edited by

Kevin Buchin
Éric Colin de Verdière



LIPICS

Editors

Kevin Buchin
Eindhoven University of Technology, Netherlands
k.a.buchin@tue.nl

Éric Colin de Verdière
CNRS, LIGM, Marne-la-Vallée, France
eric.colindeverdiere@u-pem.fr

ACM Classification 2012
Theory of computation → Computational geometry; Theory of computation → Design and analysis of algorithms; Mathematics of computing → Combinatorics; Mathematics of computing → Graph algorithms

ISBN 978-3-95977-184-9

Published online and open access by
Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany. Online available at https://www.dagstuhl.de/dagpub/978-3-95977-184-9.

Publication date
June, 2021

Bibliographic information published by the Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at https://portal.dnb.de.

Digital Object Identifier: 10.4230/LIPIcs.SoCG.2021.0

ISBN 978-3-95977-184-9 **ISSN 1868-8969** **https://www.dagstuhl.de/lipics**

LIPIcs – Leibniz International Proceedings in Informatics

LIPIcs is a series of high-quality conference proceedings across all fields in informatics. LIPIcs volumes are published according to the principle of Open Access, i.e., they are available online and free of charge.

ISSN 1868-8969

https://www.dagstuhl.de/lipics

Contents

Invited Talks

Regular Papers

37th International Symposium on Computational Geometry (SoCG 2021).
Editors: Kevin Buchin and Éric Colin de Verdière
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Media Expositions

CG Challenge Papers

Preface

The 37th International Symposium on Computational Geometry (SoCG 2021) was held online, June 7–11, 2021, as part of the Computational Geometry Week (CG Week 2021). The event was planned to take place in Buffalo, NY, USA, but eventually it was organized online because of the COVID-19 pandemic.

Altogether, 164 papers have been submitted to SoCG 2021. After a thorough review process, in which each paper has been evaluated by three or more independent reviewers, the Program Committee accepted 58 papers for presentation at SoCG 2021. These proceedings contain extended abstracts of the accepted papers, limited to 500 lines (excluding references). If any supporting material (e.g., proofs or experimental details) does not fit in the line limit, the full paper is available at a public repository and referenced in the corresponding extended abstract.

The Best Paper Award of SoCG 2021 goes to the paper "*Lower bounds for semialgebraic range searching and stabbing problems*" by Peyman Afshani and Pingan Cheng; this paper will be invited for possible publication in the *Journal of the ACM*. The Best Student Presentation Award will be determined and announced at the symposium, based on ballots cast by the attendees. A few selected papers with very positive reviews will be invited to forthcoming special issues of *Discrete & Computational Geometry* and the *Journal of Computational Geometry* dedicated to the symposium.

The SoCG Test of Time Award goes, this year, to the papers "*The analysis of a simple k-means clustering algorithm*", by Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, Angela Y. Wu, presented at SoCG 2000, and "*Constrained Delaunay triangulations*", by L. Paul Chew, presented at SoCG 1987.

The scientific program of CG Week 2021 was enriched by two distinguished invited speakers. An invited talk, entitled "*3SUM and related problems in fine-grained complexity*", was delivered by Virginia Vassilevska Williams, from the Massachusetts Institute of Technology. A second invited talk, entitled "*On Laplacians*", was delivered by Robert Ghrist, from the University of Pennsylvania. We thank these plenary speakers for kindly accepting our invitation.

In addition to the technical papers, there were two submissions to the multimedia exposition: one video and one applet. The submissions were reviewed, and both were accepted for presentation. The extended abstracts that describe these submissions are included in this proceedings volume. The multimedia content can be found at `http://www.computational-geometry.org`.

A continuing feature in this year's proceedings is the CG Challenge, now in its second year being included in the proceedings. The challenge problem this year was a coordinated motion planning problem for moving multiple labeled unit square "robots" within a regular square grid, from specified starting points to specified goal positions, in order to minimize the time when all robots have reached their goals (the makespan) or the total movement of all robots. This year there were 17 teams submitting verified solutions, and these proceedings contain contributions by the three top-placed teams describing their winning approaches.

We thank the authors of all submitted works. We are most grateful to the members of the SoCG Program Committee, the Media Exposition Committee and the CG Challenge Committee for their dedication, expertise, and hard work that ensured the high quality of the works in these proceedings. We are grateful to the assistance provided by 242 reviewers;

without their help it would be nearly impossible to run the selection process. Finally, we would like to thank Irina Kostitsyna, who kindly accepted to be the Proceedings Chair and did a meticulous work.

Many other people contributed to the success of SoCG 2021 and the entire CG Week. We are very grateful to the local organization committee for their work in organizing the event, and for switching to an online setting because of the situation. Finally, we thank all the members of the Test of Time Award, Workshop, and Young Researchers Forum Committees, the CG Challenge Advisory Board, and the Computational Geometry Steering Committee.

Kevin Buchin
SoCG program committee co-chair
Technical University Eindhoven

Éric Colin de Verdière
SoCG program committee co-chair
CNRS, LIGM, Marne-la-Vallée

Valentin Polishchuk
Media Exposition chair
Linköping University

Sándor P. Fekete
CG Challenge co-chair
TU Braunschweig

Joseph S. B. Mitchell
CG Challenge co-chair
Stony Brook University

Conference Organization

SoCG Program Committee

- Eyal Ackerman, University of Haifa at Oranim, Israel
- Pankaj K. Agarwal, Duke University, USA
- Ulrich Bauer, Technical University of Munich, Germany
- Kevin Buchin (co-chair), TU Eindhoven, Netherlands
- Jean Cardinal, Université Libre de Bruxelles, Belgium
- Éric Colin de Verdière (co-chair), CNRS, LIGM, Marne-la-Vallée, France
- Guilherme Dias da Fonseca, Aix-Marseille University, France
- Anne Driemel, Bonn University, Germany
- Ioannis Z. Emiris, NK University of Athens, and "Athena" RC, Greece
- Brittany Terese Fasy, Montana State University, USA
- Stefan Felsner, TU Berlin, Germany
- Bernd Gärtner, ETH Zurich, Switzerland
- André Lieutier, Dassault Systèmes, France
- Dmitriy Morozov Lawrence, Berkeley National Laboratory, USA
- Wolfgang Mulzer, FU Berlin, Germany
- Amir Nayyeri, Oregon State University, USA
- Jeff M. Phillips, University of Utah, USA
- Eva Rotenberg, Technical University of Denmark
- Michiel Smid, Carleton University, Canada
- Diane L. Souvaine, Tufts University, USA
- Jonathan Spreer, University of Sydney, Australia
- Konstantinos Tsakalidis, University of Liverpool, UK
- Antoine Vigneron, Ulsan National Institute of Science and Technology, Republic of Korea
- Meirav Zehavi, Ben-Gurion University, Israel

SoCG Proceedings Chair

- Irina Kostitsyna, TU Eindhoven, Netherlands

Media Exposition Committee

- Ellen Gasparovic, Union College, USA
- Roland Geraerts, Utrecht University, Netherlands
- Philipp Kindermann, University of Würzburg, Germany
- Quentin Mérigot, Université Paris-Saclay, France
- Wouter Meulemans, TU Eindhoven, Netherlands
- Valentin Polishchuk (chair), Linköping University, Sweden
- Don Sheehy, North Carolina State University, USA

CG Challenge Committee

- Sándor Fekete, TU Braunschweig, Germany
- Phillip Keldenich, TU Braunschweig, Germany
- Dominik Krupke, TU Braunschweig, Germany
- Joseph S. B. Mitchell, Stony Brook University, USA

CG Challenge Advisory Board

- William J. Cook, University of Waterloo, Canada
- Andreas Fabri, Geometry Factory, France
- Michael Kerber, Graz University of Technology, Austria
- Philipp Kindermann, University of Würzburg, Germany
- Kevin Verbeek, TU Eindhoven, Netherlands

SoCG Test of Time Award Committee

- Pankaj K. Agarwal, Duke University, USA
- Dan Halperin, Tel Aviv University, Israel
- Raimund Seidel, Saarland University, Germany

Workshop Committee

- Chao Chen, Stony Brook University, USA
- Jie Gao (chair), Rutgers University, USA
- Dan Halperin, Tel Aviv University, Israel
- Qixing Huang, The University of Texas at Austin, USA
- Irina Kostitsyna, TU Eindhoven, Netherlands
- Yusu Wang, University of California San Diego, USA

Young Researchers Forum Program Committee

- Mikkel Abrahamsen, University of Copenhagen, Denmark
- Paz Carmi, Ben-Gurion University of the Negev, Isreal
- Brittany Terese Fasy, Montana State University, USA
- Eunjin Oh, Pohang University of Science and Technology, South Korea
- Marcel Roeloffzen, TU Eindhoven, Netherlands
- Don Sheehy, North Carolina State University, USA
- Haitao Wang (chair), Utah State University, USA
- Jie Xue, University of California, Santa Barbara, USA

Local Organizing Committee

- Xin (Roger) He, University at Buffalo, The State University of New York, USA
- Ziyun Huang, Penn State Erie, The Behrend College, USA
- Shi Li, University at Buffalo, The State University of New York, USA
- Jinhui Xu (chair), University at Buffalo, The State University of New York, USA

Steering Committee (2020–2022)

- Mark de Berg (secretary), TU Eindhoven, Netherlands
- Sándor Fekete, TU Braunschweig, Germany
- Michael Hoffmann (chair), ETH Zurich, Switzerland
- Matya Katz, Ben-Gurion University of the Negev, Israel
- Bettina Speckmann, TU Eindhoven, Netherlands
- Yusu Wang (liaison with the Society for Computational Geometry), University of California San Diego, USA

Additional Reviewers

Henry Adams
Peyman Afshani
Hugo Akitaya
Greg Aloupis
Helmut Alt
Anna Arutyunova
Dominique Attali
Sayan Bandyapadhyay
Gill Barequet
Abdul Basit
Robin Belton
Helena Bergold
Ahmad Biniaz
Michael Biro
Håvard Bakke Bjerkevik
Mitchell Black
Greg Bodwin
Jean-Daniel Boissonnat
Yossi Bokor
Nicolas Bonichon
Prosenjit Bose
Magnus Bakke Botnan
Marilyn Breen
Karl Bringmann
Morten Brun
Mickaël Buchet
Benjamin A. Burton
Mathieu Carrière
Jannik Castenow
Frédéric Cazals
Parinya Chalermsook
Erin Chambers
Hsien-Chih Chang
Chao Chen
Otfried Cheong
Man-Kwun Chiu
Aruni Choudhary
Ken Clarkson
Jonas Cleve
Vincent Cohen-Addad
David Cohen-Steiner
Karthik C. S.
Justin Dallant
Hana Dal Poz Kourimska
Basudeb Datta
Minati De
Mark de Berg
Jason DeBlois
Murilo de Lima
Arnaud de Mesmay
Olivier Devillers
Tamal Dey
Charlie Dickens
Hu Ding
Vincent Divol
Mitre Dourado
Herbert Edelsbrunner
Khaled Elbassioni
Nicolas El Maalouly
Leah Epstein
Jeff Erickson
William Evans
Esther Ezra
Pan Fang
Sándor Fekete
Dan Felmdan
Henry Förster
Kyle Fox
Florian Frick
Radoslav Fulek
Zoltán Füredi
Junhao Gan
Pawel Gawrychowski
Yan Gérard
Panos Giannopoulos
Marc Glisse
Xavier Goaoc
Daniel Gonçalves
Aldo Gonzalez-Lorenzo
Sariel Har-Peled
Teresa Heiss
John Hershberger
Michael Hoffmann
Ron Holzman
Seok-Hee Hong
Alfredo Hubard
Kristóf Huszár
Klaus Jansen
Shaofeng H.-C. Jiang
Oliver Junge
Vojtěch Kaluža
Alexander Kauer
Phillip Keldenich
Chaya Keller
Michael Kerber
Minki Kim
Woojin Kim
Sándor Kisfaludi-Bak
Linda Kleist
Boris Klemz
Peter Kling
Katharina Klost
Kristin Knorr
Matias Korman
Evgenios Kornaropoulos
Irina Kostitsyna
Grigorios Koumoutsos
Martin Koutecky
Laszlo Kozma
Klaus Kriegel
Amer Krivosija
Nirman Kumar
Jean-Philippe Labbé
Claudia Landi
Stefan Langerman
Elmar Langetepe
Francis Lazarus
Hung Le
Fabian Lenzen
Michael Lesnick
David Letscher
Jacob Leygonie
Luc Libralesso
Anita Liebenau
Maarten Löffler
Anna Lubiw
Benjamin Lund
Anil Maheshwari
Huy Mai
Samuel Mansfield
Clément Maria
Victor Marsault

37th International Symposium on Computational Geometry (SoCG 2021).
Editors: Kevin Buchin and Éric Colin de Verdière
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Tomáš Masařík
William Maxwell
Brad McCoy
Saeed Mehrabi
Samuel Micka
David L. Millman
Till Miltzow
Joseph Mitchell
Nicolas Montana
Pat Morin
David Mount
Elizabeth Munch
Meiram Murzabulatov
Christopher Musco
Maryam Negahbani
Frank Nielsen
Arnur Nigmetov
Bengt J. Nilsson
André Nusser
Ippei Obayashi
Johannes Obenaus
Karolina Okrasa
Deborah Oliveros
Dömötör Pálvölgyi
Evanthia Papadopoulou
Salman Parsa
Pavel Paták
Zuzana Patáková
Amit Patel
Balazs Patkos
Yuval Peled
Marco Pellegrini
Martin Pergel
Ioannis Psarros
Jessica Purcell
Sharath Raghvendra
Benjamin Raichel
Sébastien Ratel
Abhishek Rathod
Jürgen Richter-Gebert
Bastian Rieck
Vanessa Robins
Dennis Rohde
Alexander Rolle
Günter Rote
Atri Rudra
Rik Sarkar
Anna Schenfisch
Manfred Scheucher
Stefan Schirra
Lena Schlipf
Patrick Schnider
Rafael Schouery
Hannah Schreiber
Felix Schröder
André Schulz
Jordan Schupbach
Luis Scoccola
Martina Scolamiero
Carlos Seara
Eric Sedgwick
C. Seshadhri
Martin P. Seybold
Mordechai Shalom
Jonathan Shewchuk
Anastasios Sidiropoulos
Rodrigo Silveira
Stavros Sintos
Primoz Skraba
Kelly Spendlove
Boris Springborn
Raphael Steiner
Sebastian Urban Stich
Andrew Suk
Edward Swartz
Martin Tancer
Erin Taylor
Monique Teillaud
Csaba Tóth
Geza Tóth
Kostas Tsichlas
Katharine Turner
Hemant Tyagi
Torsten Ueckerdt
Jalaj Upadhyay
Marc van Kreveld
André van Renssen
Mikael Vejdemo-Johansson
Santosh Vempala
Suresh Venkatasubramanian
Mate Vizer
Julien Vuillamy
Hubert Wagner
Zhengchao Wan
Yusu Wang
Guowei Wei
Oren Weimann
Emo Welzl
Rephael Wenger
Andreas Wiese
Sebastian Wild
Max Willert
Mathijs Wintraecken
Sampson Wong
Matthew Wright
Marcin Wrochna
Jinhui Xu
Ling Zhou
Binhai Zhu

On Laplacians

Robert Ghrist
University of Pennsylvania, Philadelphia, PA, USA

Abstract

The graph Laplacian is a fundamental tool in computational geometry and data science; its geometric and spectral properties combine to yield a bridge between analytic methods, geometric intuition, and topological properties.

This talk will survey recent generalizations of the graph Laplacian inspired by algebraic geometry and algebraic topology. Given a sheaf over a graph – a functor from the vertex-edge poset to a category of algebraic data – one can assign a type of Hodge Laplacian which, in the case of a constant sheaf of 1-dimensional vector spaces, recapitulates the classic graph Laplacian. This Hodge (or *sheaf*) Laplacian is the basis for diffusion and distributed algorithms over network sheaves. This talk has very few prerequisites and will introduce the concepts carefully in the context of applications ranging from distributed optimization, to learning, opinion dynamics, neural networks, and more.

From a mathematical perspective, the most interesting challenge is the extension of Laplacians to systems – or sheaves – taking values in non-abelian categories, such as categories of lattices. Such extensions will be a particular focus of the talk.

This talk surveys joint works with Jakob Hansen, Paige Randall North, and Hans Riess.

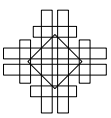
2012 ACM Subject Classification Mathematics of computing → Algebraic topology

Keywords and phrases Laplacian, sheaf theory, applied topology

Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.1

Category Invited Talk

Funding *Robert Ghrist*: This work was funded by the Office of the Assistant Secretary of Defense Research & Engineering through a Vannevar Bush Faculty Fellowship, ONR N00014-16-1-2010.



3SUM and Related Problems in Fine-Grained Complexity

Virginia Vassilevska Williams
Massachusetts Institute of Technology, Cambridge, MA, USA

Abstract

3SUM is a simple to state problem: given a set S of n numbers, determine whether S contains three a, b, c so that $a + b + c = 0$. The fastest algorithms for the problem run in $n^2\text{poly}(\log\log n)/(\log n)^2$ time both when the input numbers are integers [1] (in the word RAM model with $O(\log n)$ bit words) and when they are real numbers [2] (in the real RAM model).

A hypothesis that is now central in Fine-Grained Complexity (FGC) states that 3SUM requires $n^{2-o(1)}$ time (on the real RAM for real inputs and on the word RAM with $O(\log n)$ bit numbers for integer inputs). This hypothesis was first used in Computational Geometry by Gajentaan and Overmars [4] [1] who built a web of reductions showing that many geometric problems are hard, assuming that 3SUM is hard. The web of reductions within computational geometry has grown considerably since then (see some citations in [11]).

A seminal paper by Pătraşcu [7] showed that the integer version of the 3SUM hypothesis can be used to prove polynomial conditional lower bounds for several problems in data structures and graph algorithms as well, extending the implications of the hypothesis to outside computational geometry. Pătraşcu proved an important tight equivalence between (integer) 3SUM and a problem called 3SUM-Convolution (see also [3]) that is easier to use in reductions: given an integer array a of length n, do there exist $i, j \in [n]$ so that $a[i] + a[j] = a[i + j]$. From 3SUM-Convolution, many 3SUM-based hardness results have been proven: e.g. to listing graphs in triangles, dynamically maintaining shortest paths or bipartite matching, subset intersection and many more.

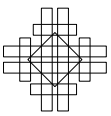
It is interesting to consider more runtime-equivalent formulations of 3SUM, with the goal of uncovering more relationships to different problems. The talk will outline some such equivalences. For instance, 3SUM (over the reals or the integers) is equivalent to All-Numbers-3SUM: given a set S of n numbers, determine for every $a \in S$ whether there are $b, c \in S$ with $a + b + c = 0$ (e.g. [10]).

The equivalences between 3SUM, 3SUM-Convolution and All-Numbers 3SUM are (n^2, n^2)-fine-grained equivalences that imply that if there is an $O(n^{2-\varepsilon})$ time algorithm for one of the problems for $\varepsilon > 0$, then there is also an $O(n^{2-\varepsilon'})$ time algorithm for the other problems for some $\varepsilon' > 0$. More generally, for functions $a(n), b(n)$, there is an (a, b)-fine-grained reduction [11, 9, 10] from problem A to problem B if for every $\varepsilon > 0$ there is a $\delta > 0$ and an $O(a(n)^{1-\delta})$ time algorithm for A that does oracle calls to instances of B of sizes $n_1, \ldots, n_k$ (for some k) so that $\sum_{j=1}^{k} b(n_j)^{1-\varepsilon} \leq a(n)^{1-\delta}$. With such a reduction, an $O(b(n)^{1-\varepsilon})$ time algorithm for B can be converted into an $O(a(n)^{1-\delta})$ time algorithm for A by replacing the oracle calls by calls to the B algorithm. A and B are (a, b)-fine-grained equivalent if A (a, b)-reduces to B and B (b, a)-reduces to A.

One of the main open problems in FGC is to determine the relationship between 3SUM and the other central FGC problems, in particular All-Pairs Shortest Paths (APSP). A classical graph problem, APSP in n node graphs has been known to be solvable in $O(n^3)$ time since the 1950s. Its fastest known algorithm runs in $n^3/exp(\sqrt{\log n})$ time [14]. The APSP Hypothesis states that $n^{3-o(1)}$ time is needed to solve APSP in graphs with integer edge weights in the word-RAM model with $O(\log n)$ bit words. It is unknown whether APSP and 3SUM are fine-grained reducible to each other, in either direction. The two problems are very similar. Problems such as $(\min, +)$-convolution (believed to require $n^{2-o(1)}$ time) have tight fine-grained reductions to both APSP and 3SUM, and both 3SUM and APSP have tight fine-grained reductions to problems such as Exact Triangle [10, 8, 12] and (since very recently) Listing triangles in sparse graphs [7, 6, 13]. The talk will discuss these relationships and some of their implications, e.g. to dynamic algorithms.

[1] They used a more stringent version of the hypothesis that said that (real) 3SUM requires $\Omega(n^2)$ time which we now know to be false [5].

37th International Symposium on Computational Geometry (SoCG 2021).
Editors: Kevin Buchin and Éric Colin de Verdière; Article No. 2; pp. 2:1–2:2



Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

2012 ACM Subject Classification Theory of computation; Theory of computation → Design and analysis of algorithms

Keywords and phrases fine-grained complexity

Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.2

Category Invited Talk

References

1 Ilya Baran, Erik D. Demaine, and Mihai Pătraşcu. Subquadratic algorithms for 3sum. In *Algorithms and Data Structures, 9th International Workshop, WADS 2005, Waterloo, Canada, August 15-17, 2005, Proceedings*, pages 409–421, 2005.

2 Timothy M. Chan. More logarithmic-factor speedups for 3sum, (median, +)-convolution, and some geometric 3sum-hard problems. *ACM Trans. Algorithms*, 16(1):7:1–7:23, 2020.

3 Timothy M. Chan and Qizheng He. Reducing 3sum to convolution-3sum. In Martin Farach-Colton and Inge Li Gørtz, editors, *3rd Symposium on Simplicity in Algorithms, SOSA 2020, Salt Lake City, UT, USA, January 6-7, 2020*, pages 1–7. SIAM, 2020.

4 A. Gajentaan and M. Overmars. On a class of $O(n^2)$ problems in computational geometry. *Computational Geometry*, 5(3):165–185, 1995.

5 Allan Grønlund and Seth Pettie. Threesomes, degenerates, and love triangles. *J. ACM*, 65(4):22:1–22:25, 2018.

6 Tsvi Kopelowitz, Seth Pettie, and Ely Porat. Higher lower bounds from the 3sum conjecture. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10–12, 2016*, pages 1272–1287, 2016.

7 Mihai Pătraşcu. Towards polynomial lower bounds for dynamic problems. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5–8 June 2010*, pages 603–610, 2010.

8 V. Vassilevska and R. Williams. Finding, minimizing, and counting weighted subgraphs. In *Proc. STOC*, pages 455–464, 2009.

9 V. Vassilevska Williams and R. Williams. Subcubic equivalences between path, matrix and triangle problems. In *Proc. FOCS*, pages 645–654, 2010.

10 V. Vassilevska Williams and R. Williams. Subcubic equivalences between path, matrix and triangle problems. *Journal of the ACM*, 2018. to appear.

11 Virginia Vassilevska Williams. On some fine-grained questions in algorithms and complexity. In *Proceedings of the International Congress of Mathematicians (ICM 2018)*, pages 3447–3487, 2018. doi:10.1142/9789813272880_0188.

12 Virginia Vassilevska Williams and Ryan Williams. Finding, minimizing, and counting weighted subgraphs. *SIAM J. Comput.*, 42(3):831–854, 2013.

13 Virginia Vassilevska Williams and Yinzhan Xu. Monochromatic triangles, triangle listing and APSP. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16–19, 2020*, pages 786–797. IEEE, 2020.

14 Ryan Williams. Faster all-pairs shortest paths via circuit complexity. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31–June 03, 2014*, pages 664–673, 2014.

Classifying Convex Bodies by Their Contact and Intersection Graphs

Anders Aamand
BARC, University of Copenhagen, Denmark

Mikkel Abrahamsen
BARC, University of Copenhagen, Denmark

Jakob Bæk Tejs Knudsen
BARC, University of Copenhagen, Denmark

Peter Michael Reichstein Rasmussen
BARC, University of Copenhagen, Denmark

Abstract

Let A be a convex body in the plane and $A_1, \ldots, A_n$ be translates of A. Such translates give rise to an *intersection graph* of A, $G = (V, E)$, with vertices $V = \{1, \ldots, n\}$ and edges $E = \{uv \mid A_u \cap A_v \neq \emptyset\}$. The subgraph $G' = (V, E')$ satisfying that $E' \subset E$ is the set of edges uv for which the interiors of A_u and A_v are disjoint is a *unit distance* graph of A. If furthermore $G' = G$, i.e., if the interiors of A_u and A_v are disjoint whenever $u \neq v$, then G is a *contact graph* of A.

In this paper, we study which pairs of convex bodies have the same contact, unit distance, or intersection graphs. We say that two convex bodies A and B are equivalent if there exists a linear transformation B' of B such that for any slope, the longest line segments with that slope contained in A and B', respectively, are equally long. For a broad class of convex bodies, including all strictly convex bodies and linear transformations of regular polygons, we show that the contact graphs of A and B are the same if and only if A and B are equivalent. We prove the same statement for unit distance and intersection graphs.

2012 ACM Subject Classification Theory of computation → Computational geometry; Mathematics of computing → Graph theory; Mathematics of computing → Discrete mathematics

Keywords and phrases convex body, contact graph, intersection graph

Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.3

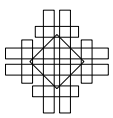
Related Version *Full Version*: `https://arxiv.org/abs/1902.01732` [1]

Funding The authors are part of BARC, Basic Algorithms Research Copenhagen, supported by the VILLUM Foundation grant 16582.

Acknowledgements We thank Tillmann Miltzow for asking when the translates of two different convex bodies induce the same intersection graphs which inspired us to work on these problems.

1 Introduction

Consider a convex body A, i.e., a convex, compact region of the plane with non-empty interior, and let $\mathcal{A} = \{A_1, \ldots, A_n\}$ be a set of n translates of A. Then $\mathcal{A}$ gives rise to an *intersection graph* $G = (V, E)$, where $V = \{1, \ldots, n\}$ and $E = \{uv \mid A_u \cap A_v \neq \emptyset\}$, and a *unit distance graph* $G' = (V, E')$, where $uv \in E'$ if and only if $uv \in E$ and A_u and A_v have disjoint interiors. In the special case that $G = G'$ (i.e., the convex bodies of $\mathcal{A}$ have pairwise disjoint interiors), we say that G is a *contact graph* (also known as a *touch graph* or *tangency graph*). Thus, A defines three classes of graphs, namely the intersection graphs $I(A)$, the unit distance graphs $U(A)$, and the contact graphs $C(A)$ of translates of A.



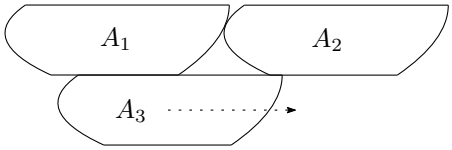


Figure 1 Translates of a convex body not having the URTC property. The disk A_3 can "slide" along A_1 and A_2.

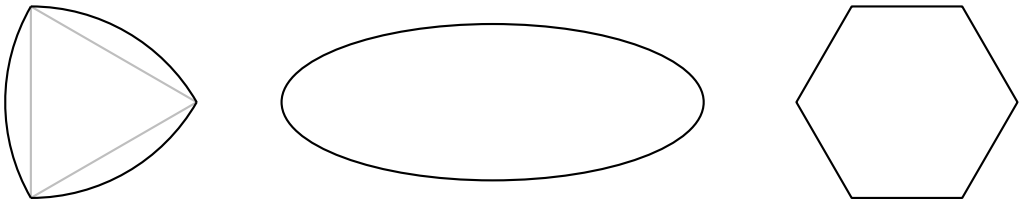


Figure 2 Reuleaux triangle (left), ellipse (middle), and regular hexagon (right).

The study of intersection graphs has been an active research area in discrete and computational geometry for the past three decades. For instance, numerous papers consider the problem of solving classical graph problems efficiently on various classes of geometric intersection graphs; see Section 1.1 for some references. Meanwhile, the study of contact graphs of translates of a convex body has much older roots. It is closely related to the packings of such a body, which has a very long and rich history in mathematics going back (at least) to the seventeenth century, where research on the packings of circles of varying and constant radii was conducted and Kepler famously conjectured upon a 3-dimensional counterpart of such problems, the packing of spheres.

In this paper we investigate the question of when two convex bodies A and B give rise to the same classes of graphs. We restrict ourselves to convex bodies A that have the URTC property (*unique regular triangle constructibility*). This is the property that given two interior disjoint translates A_1, A_2 of A that touch, there are exactly two ways to place a third translate A_3 such that A_3 is interior disjoint from A_1 and A_2, but touches both. Convex bodies with the URTC property include all linear transformations of regular polygons except squares and all strictly convex bodies [15]. Thus, almost all convex bodies in a measure theoretic sense have the property [15, 18, 30]. A convex body without the property must have a sufficiently long line segment on the boundary (to be made precise in Section 1.3); see Figure 1 for an example.

The main result of the paper is summarized in the following theorem.

▶ **Theorem 1.** *Let A and B be convex bodies with the URTC property. Then each of the identities $I(A) = I(B)$, $U(A) = U(B)$, and $C(A) = C(B)$ holds if and only if the following condition is satisfied: there is a linear transformation B' of B such that for any slope, the longest segments contained in A and B', respectively, with that slope are equally long.*

If the condition from the theorem is satisfied, we say that A and B are *equivalent*. The length of the longest segment with a given slope contained in a convex body A is often called the *width* of A in the corresponding direction. A circle has constant width but there are other convex bodies of constant width, the simplest example being the Reuleaux triangle; see Figure 2. As an example it follows from the theorem that circles and Reuleaux triangles have the same contact, unit distance, and intersection graphs, which in turn are the same as those for ellipses (ellipses are linear transforms of circles). It also follows that these classes are different from those of regular hexagons.

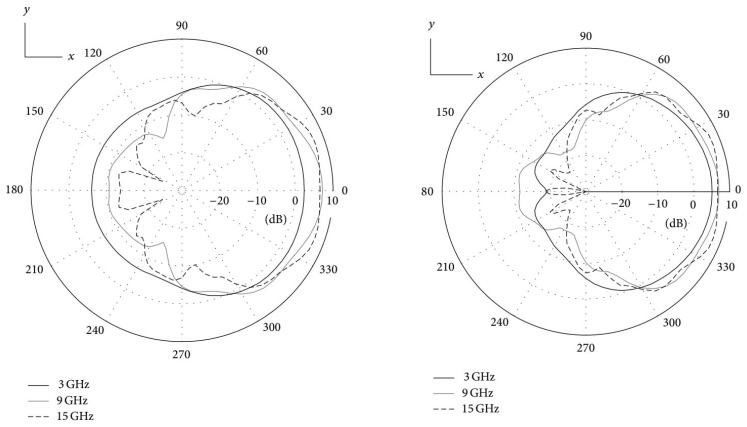


Figure 3 The strength of radiation in every direction and at various frequencies for two different transmitters described in [25]. In engineering circles, this known as the *radiation pattern*.

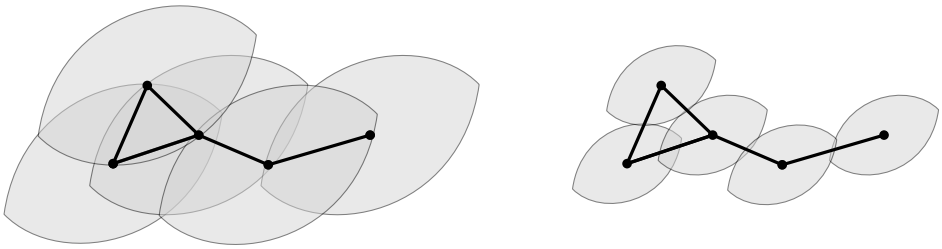


Figure 4 When the reachable region of a device is symmetric and the devices are oriented in the same way, a communication network is the intersection graph of the reachable region scaled by 1/2. Left: A network of five identical devices with the reachable regions shown. Right: The intersection graph of the reachable regions scaled by 1/2.

1.1 Practical Implications

From a practical point of view, the research on intersection graphs is often motivated by the applicability of these graphs when modeling wireless communication networks and facility location problems. If a device is located at some point in the plane and is able to transmit to and receive from all other devices within some distance, then the devices can be represented as unit disks in such a way that two devices can communicate if and only if their disks overlap. Many highly-cited papers gave this motivation for studying unit disk intersection graphs [9, 14, 16, 21, 29] and it remains a motivation for new research [7, 12, 13, 24].

However, it is in general not the case that a transmitter emits an equally strong signal in all directions. For a real-world example of how the signal strength may vary in different directions; see Figure 3. If the networks that can be made with devices of a given type are not the unit disk intersection graphs, the algorithms for unit disk graphs cannot be expected to work when applied to the actual networks. It is therefore necessary to study how the possible networks that can be made with devices of different types depend on the radiation pattern of the devices. See Figure 4 for a demonstration of the connection between communication networks of a device with a non-circular radiation pattern and intersection graphs of the corresponding convex body.

1.2 Other Related Work

An important notion in the area of contact graphs is that of the *Hadwiger number* of a body K, which is the maximum possible number of pairwise interior-disjoint translates K_i of K that each touch but do not overlap K. The Hadwiger number of K is thus the maximum degree of a contact graph of translates of K. In the plane, the Hadwiger number is 8 for parallelograms and 6 for all other convex bodies. We refer the reader to the books and surveys by László and Gábor Fejes Tóth [28, 10] and Böröczky [3].

Another noteworthy result on contact graphs is the Circle Packing Theorem (also known as the Koebe–Andreev–Thurston Theorem): A graph is simple and planar if and only if it is the contact graph of some set of circular disks in the plane (the radii of which need not be equal). The result was proven by Koebe in 1935 [19] (see [11] for a streamlined, elementary proof). Schramm [26] generalized the circle packing theorem by showing that if a planar convex body with smooth boundary is assigned to each vertex in a planar graph, then the graph can be realized as a contact graph where each vertex is represented by a homothet (i.e., a scaled translation) of its assigned body.

Several papers have compared classes of intersection graphs of various geometric objects, see for instance [4, 6, 8, 17, 20]. Most of the results are inclusions between classes of intersection graphs of one-dimensional objects such as line segments and curves.

A survey by Swanepoel [27] summarizes results on minimum distance graphs and unit distance graphs in normed spaces, including bounds on the minimum/maximum degree, maximum number of edges, chromatic number, and independence number.

In the area of computational geometry, Müller, van Leeuwen, and van Leeuwen [23] gave sharp upper and lower bounds on the size of an integer grid used to represent an intersection graph of translates of a convex polygon with corners at rational coordinates. Their results imply that for any convex polygon R with rational corners, the problem of recognizing intersection graphs of translates of R is in NP. On the contrary, it is open whether recognition of unit disk intersection graphs in the Euclidean plane is in NP. Indeed, the problem is $\exists\mathbb{R}$-complete (and thus in PSPACE), and using integers to represent the center coordinates and radii of the disks in some graphs requires exponentially many bits [5, 22].

Bonnet, Grelier, and Miltzow [2] showed how well-known algorithms for the clique problem in unit disk intersection graphs and disk intersection graphs can be adjusted to work for intersection graphs of translates or homothets of an arbitrary centrally symmetric convex body.

1.3 Preliminaries

We begin by defining some basic geometric concepts and terminology.

For a subset $A \subset \mathbb{R}^2$ of the plane we denote by A° the interior of A, that is,

$$A^\circ = \{x \in A \mid \exists \text{ open } U \subset \mathbb{R}^2 \text{ such that } \{x\} \subset U \subset A\}.$$

We say that A is a *convex body* if A is compact, convex, and has non-empty interior. We say that A is *symmetric* if whenever $x \in A$, then $-x \in A$. It is well-known that if A is a symmetric convex body, then the map $\|\cdot\|_A : \mathbb{R}^2 \to \mathbb{R}_{\geq 0}$ defined by

$$\|x\|_A = \inf\{\lambda \geq 0 \mid x \in \lambda A\},$$

is a norm. Moreover $A = \{x \in \mathbb{R}^2 \mid \|x\|_A \leq 1\}$ and $A^\circ = \{x \in \mathbb{R}^2 \mid \|x\|_A < 1\}$.

It follows from these properties that for translates $A_1 = A + v_1$ and $A_2 = A + v_2$ it holds that $A_1 \cap A_2 \neq \emptyset$ if and only if $\|v_1 - v_2\|_A \leq 2$ and $A_1^\circ \cap A_2^\circ \neq \emptyset$ if and only if $\|v_1 - v_2\|_A < 2$. This means that when studying contact, unit distance, and intersection

graphs of a symmetric convex body A, we can shift our viewpoint from translates of A to point sets in $\mathbb{R}^2$ and their $\|\cdot\|_A$-distances: If $\mathcal{A} \subset \mathbb{R}^2$ is a set of points we define $I_A(\mathcal{A})$ and $U_A(\mathcal{A})$ to be the graphs with vertex set $\mathcal{A}$ and edge sets $\{(x,y) \in \mathcal{A}^2 \mid x \neq y \text{ and } \|x-y\|_A \leq 2\}$ and $\{(x,y) \in \mathcal{A}^2 \mid \|x-y\|_A = 2\}$, respectively. Moreover, if for all distinct points $x, y \in \mathcal{A}$ it holds that $\|x-y\|_A \geq 2$, we say that $\mathcal{A}$ is *compatible with* A and define $C_A(\mathcal{A})$ to be the graph with vertex set $\mathcal{A}$ and edge set $\{(x,y) \in \mathcal{A}^2 \mid \|x-y\|_A = 2\}$. Then $I_A(\mathcal{A}), U_A(\mathcal{A})$, and $C_A(\mathcal{A})$, respectively, are isomorphic to the intersection, unit distance, and contact graph of A realized by the translates $(A+a)_{a\in\mathcal{A}}$. When studying contact, unit distance, and intersection graphs of a symmetric, convex body A we will view them as being induced by point sets rather than by translates of A.

We say that a (not necessarily symmetric) convex body A in the plane has the URTC property if the following holds: For any two interior disjoint translates of A, A_1 and A_2, satisfying $A_1 \cap A_2 \neq \emptyset$, there exists precisely two vectors $v \in \mathbb{R}^2$ such that for $i \in \{1,2\}$, $(A+v)^\circ \cap A_i^\circ = \emptyset$ but $(A+v) \cap A_i \neq \emptyset$. If A is symmetric, this amounts to saying that for any two points $v_1, v_2 \in \mathbb{R}^2$ with $\|v_1 - v_2\|_A = 2$, the set $\{v \in \mathbb{R}^2 \mid \|v - v_1\|_A = \|v - v_2\|_A = 2\}$ has size two. Gehér [15] proved that a symmetric convex body A has the URTC property if and only if the boundary ∂A does not contain a line segment of length more than 1 in the $\|\cdot\|_A$-norm. See Figure 1 for an example of a convex body not having the URTC property.

A *drawing* of a graph $G \in I(A)$ as an intersection graph of a symmetric convex body A is a point set $\mathcal{A} \subset \mathbb{R}^2$ and a set of straight line segments $\mathcal{L}$ such that $I_A(\mathcal{A})$ is isomorphic to G and $\mathcal{L}$ is exactly the line segments between the points $u, v \in \mathcal{A}$ which are connected by an edge in G. We define a drawing of a graph G as a contact and unit distance graph similarly.

For a norm $\|\cdot\|$ on $\mathbb{R}^2$ and a line segment ℓ with endpoints a and b we will often write $\|\ell\| = \|ab\|$ instead of $\|a-b\|$. Also, if A is a symmetric convex body and $U, V \subset \mathbb{R}^2$, we define $d_A(U,V) := \inf\{\|uv\|_A \mid (u,v) \in U \times V\}$.

1.4 Structure and Techniques of the Paper

Establishing the sufficiency of the condition of Theorem 1, i.e., showing that if A and B are equivalent then $I(A) = I(B)$, $U(A) = U(B)$, and $C(A) = C(B)$, is relatively straightforward and has been deferred to the full version of the paper. It is also relatively easy to reduce Theorem 1 to the case where the convex bodies are symmetric so this too is deferred to the full version. When both A and B are symmetric, they are equivalent according to the condition of Theorem 1 if and only if they are linear transformations of each other.

Thus, left with the task of proving the necessity of the condition of Theorem 1 in the symmetric case, we proceed in two steps. First, in Section 2, we prove the following result, which for contact and unit distance graphs is a generalization of this direction of Theorem 1.

▶ **Theorem 2.** *Let A and B be symmetric convex bodies with the URTC property such that A is not a linear transformation of B. There exists a graph $G \in C(A)$ such that for all $H \in C(B)$ and all subgraphs $H' \subseteq H$, G is not isomorphic to H'. In particular $C(A) \setminus C(B) \neq \emptyset$.*

As we will also discuss in Section 2 the same result holds if $C(X)$ is replaced by $U(X)$ for $X \in \{A, B\}$ everywhere in the theorem above and the proof is identical.

The core idea in proving Theorem 2 is to consider a graph G satisfying that any drawing of G as a contact graph of A has certain structural properties. Concretely, we ensure that any drawing of G as a contact graph of A consists of many large hollow hexagons. In the interior of each hexagon, we force there to be a "bridge" of translates of A connecting the

sides of the hexagon. We show that if B is not a linear transformation of A, then the contact graph cannot be realized by translates of B if we make sufficiently many and sufficiently large hexagons with bridges of different slopes. See Figures 6 and 7 for illustrations.

To include intersection graphs, we proceed with the second step. In Section 3, we prove the following result which combined with Theorem 2 immediately yields the necessity of the condition of Theorem 1 for intersection graphs.

▶ **Theorem 3.** *Let A and B be symmetric convex bodies. If there exists a graph $G \in C(A)$ such that for all $H \in C(B)$ and all subgraphs $H' \subset H$, G is not isomorphic to H', then $I(A) \neq I(B)$.*

This result holds for general symmetric convex bodies. An improvement of Theorem 2 to general symmetric convex bodies (not necessarily having the URTC property) would thus yield a version of Theorem 1 that also holds for general convex bodies.

In order to prove Theorem 3, we proceed as follows. For every positive integer k we construct a gadget $Q_k \in I(A)$ which contains as a subgraph a distinguished cycle $\alpha_k \subset Q_k$. We prove that in any drawing of Q_k as an intersection graph of translates of A, α_k is contained in a translation of the annulus $kA \setminus (k-1)A$ (here, $kA = \{ka \mid a \in A\}$ is the scaling of A by k). This allows us to view α_k as an upscaled copy of the boundary of A with a precision error decreasing in k. Similarly, in any drawing of the same gadget Q_k as an intersection graph of another body, B, the cycle α_k appears as an upscaled copy of B. The idea is then to simulate a contact graph $G \in C(A)$ using distinct copies of Q_k, where each copy plays the role of a single vertex in G. If A is not a linear transformation of B, we can choose G with the properties promised in Theorem 2. We are then able to prove that if we choose k sufficiently large (i.e., obtaining sufficiently high resemblence between α_k and an upscaled copy of A resp. B), then we can realize the simulation of G as an intersection graph using translates of A, but not using translates of B. This then implies $I(A) \neq I(B)$ as desired.

Beyond aiding us in the proof of our main theorem, we believe that this proof technique – the reduction from intersection to contact graphs – is of independent interest. It appears a novel approach with the potential to answer other questions on intersection graphs.

2 Contact and Unit Distance Graphs

In this section we prove Theorem 2. The proof for unit distance graphs is completely identical so we will merely provide a remark justifying this claim by the end of the section.

Throughout the section A and B will denote symmetric convex bodies. For $\theta \in [0, 2\pi)$ we define $e_A(\theta)$ to be the vector of argument θ and with $\|e_A(\theta)\|_A = 1$. We also define $\rho_A(\theta) = 2\|e_A(\theta)\|_2$. Then $\rho_A(\theta)$ can be thought of as the "diameter" of A in direction θ. One of our most important tools is the following lemma.

▶ **Lemma 4.** *Let A, B be symmetric convex bodies in $\mathbb{R}^2$. If for every finite set $\Theta \subset [0, \pi)$ and for every $\varepsilon > 0$, there exists a linear map $T\colon \mathbb{R}^2 \to \mathbb{R}^2$ satisfying that $|\rho_{T(B)}(\theta) - \rho_A(\theta)| < \varepsilon$ for all $\theta \in \Theta$, then there exists a linear map $T\colon \mathbb{R}^2 \to \mathbb{R}^2$ with $T(B) = A$.*

Due to space limitations we have left out the proof, but it can be found in the full version.

We proceed to describe certain lattices which give rise to contact graphs that can only be realized in an essentially unique way. We start with the following definition.

▶ **Definition 5.** *Let $A \subset \mathbb{R}^2$ be a symmetric convex body, and $\|\cdot\|_A$ the associated norm. Let $e_1, e_2 \in \mathbb{R}^2$ be such that $\|e_1\|_A = \|e_2\|_A = \|e_1 - e_2\|_A = 2$. We define the lattice $\mathcal{L}_A(e_1, e_2) = \{a_1e_1 + a_2e_2 \mid (a_1, a_2) \in \mathbb{Z}^2\}$.*

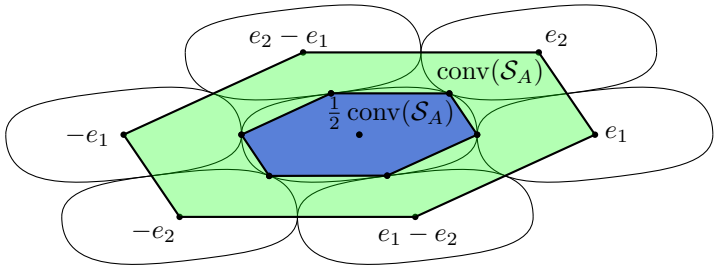


Figure 5 A symmetric convex body A and the sets $\frac{1}{2}\operatorname{conv}(\mathcal{S}_A)$ and $\operatorname{conv}(\mathcal{S}_A)$ (blue and green respectively) which satisfies $\frac{1}{2}\operatorname{conv}(\mathcal{S}_A) \subseteq A \subseteq \operatorname{conv}(\mathcal{S}_A)$.

The left hand side of figure Figure 6 illustrates the lattice structure. Let us assume that A has the URTC property and describe a few properties of the lattice $\mathcal{L}_A(e_1, e_2)$. After choosing e_1 with $\|e_1\|_A = 2$, there are precisely two vectors v with $\|v\|_A = \|v - e_1\|_A = 2$, using the URTC property. If one is v_2 the second is $e_1 - v_2$ so regardless how we choose e_2 we obtain the same lattice. Using the triangle inequality and the URTC property of A it is easily verified that for distinct $x, y \in \mathcal{L}_A(e_1, e_2)$, $\|x - y\|_A \geq 2$ with equality holding exactly if $x - y \in \mathcal{S}_A := \{e_1, e_2, e_2 - e_1, -e_1, -e_2, e_1 - e_2\}$. This implies that the contact graph $G_0 := C_A(\mathcal{L}_A(e_1, e_2))$ is in fact (isomorphic to) an infinite triangular grid.

Another useful fact is the following:

▶ **Lemma 6.** *With $\mathcal{S}_A$ as above it holds that $\frac{1}{2}\operatorname{conv}(\mathcal{S}_A) \subset A \subset \operatorname{conv}(\mathcal{S}_A)$. Here $\operatorname{conv}(\mathcal{S}_A)$ is the convex hull of $\mathcal{S}_A$. If in particular B is another symmetric convex body for which $\|e_1\|_B = \|e_2\|_B = \|e_1 - e_2\|_B = 2$, then for all $x \in \mathbb{R}^2$ it holds that $\frac{1}{2}\|x\|_A \leq \|x\|_B \leq 2\|x\|_A$.*

Proof. See Figure 5. As $\frac{1}{2}\mathcal{S}_A \subset A$ and A is convex the first inclusion is clear. For the second inclusion we note that all points y on the hexagon connecting the points $e_1, e_2, e_2 - e_1, -e_1, -e_2, e_1 - e_2$ of $\mathcal{S}_A$ in this order have $\|y\|_A \geq 1$ by the triangle inequality and so $A \subset \operatorname{conv}(\mathcal{S}_A)$.

For the last statement of the lemma note that if $x \in \mathbb{R}^2$ then

$$\|x\|_B \geq \inf_{\lambda \geq 0}\{x \in \lambda \operatorname{conv}(\mathcal{S}_B)\} = \inf_{\lambda \geq 0}\{x \in \lambda \operatorname{conv}(\mathcal{S}_A)\} \geq \inf_{\lambda \geq 0}\{x \in 2\lambda A\} = \frac{1}{2}\|x\|_A,$$

and similarly $\|x\|_A \geq \frac{1}{2}\|x\|_B$. ◀

▶ **Definition 7.** *We say that a graph $G = (V, E)$ is* lattice unique *if $|V| = n \geq 3$ and there exists an enumeration of its vertices $v_1, \ldots, v_n$ such that*

- *The vertex induced subgraph $G[v_1, v_2, v_3] \simeq K_3$ is a triangle.*
- *For $i > 3$ there exists distinct $j, k, l < i$ such that $G[v_j, v_k, v_l] \simeq K_3$ and both (v_i, v_j) and (v_i, v_k) are edges of G.*

Suppose that A is a symmetric convex body with the URTC property, that $\mathcal{A} \subset \mathbb{R}^2$ is compatible with A, and that $G = C_A(\mathcal{A})$ is lattice unique. Enumerate the points of $\mathcal{A} = \{v_1, \ldots, v_n\}$ according to the definition of lattice uniqueness. Without loss of generality assume that $v_1 = 0$. Then the URTC property of A combined with the lattice uniqueness of G gives that $v_4, \ldots, v_n$ are uniquely determined from v_2 and v_3 and all contained in $\mathcal{L}_A(v_2, v_3)$. If moreover B is another convex body with the URTC property, $\mathcal{B} = \{v_1', \ldots, v_n'\} \subset \mathbb{R}^2$ has $v_1' = 0$ and is compatible with B, and $C_B(\mathcal{B}) \simeq C_A(\mathcal{A})$ via the graph isomorphism $\varphi : v_i' \mapsto v_i$, then the linear map $T : \mathbb{R}^2 \to \mathbb{R}^2$ defined by $T : a_1 v_2' + a_2 v_3' \mapsto a_1 v_2 + a_2 v_3$ satisfies that $T|_{\mathcal{B}} = \varphi$.

We will use this observation in the proof of Theorem 2 which we now provide.

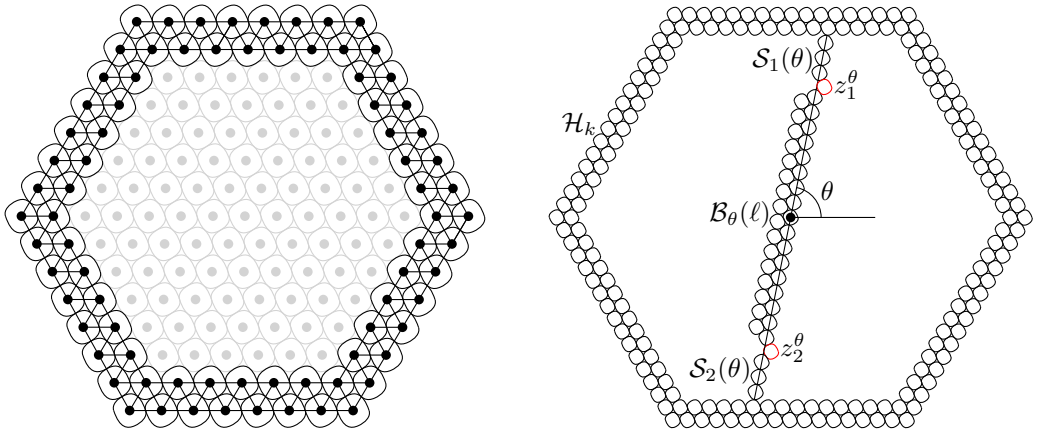


Figure 6 Left: The points of $\mathcal{H}_6$ along with the corresponding lattice unique subgraph $G_0[\mathcal{H}_6]$. Right: The attachment of the beam $\mathcal{B}_\theta(\ell)$.

Proof of Theorem 2. We choose $e_1, e_2 \in \mathbb{R}^2$ be such that $\|e_1\|_A = \|e_2\|_A = \|e_1 - e_2\|_A = 2$ and define the lattice $\mathcal{L} := \mathcal{L}_A(e_1, e_2)$. We also define the infinite graph $G_0 := C_A(\mathcal{L})$ which by the remarks following Definition 5 is isomorphic to the infinite triangular grid. Without loss of generality we can assume that e_1 and e_2 satisfy that $\|e_1\|_2 = \|e_2\|_2 = \|e_1 - e_2\|_2 = 2$, since there exists a non-singular linear transformation T such that $\|T(e_1)\|_2 = \|T(e_2)\|_2 = \|T(e_1) - T(e_2)\|_2 = 2$, and $C(A) = C(T(A))$. Note that in this setting we can use Lemma 6 to compare A to the disk of radius 1 and obtain $\frac{1}{2}\|x\|_2 \leq \|x\|_A \leq 2\|x\|_2$ for every $x \in \mathbb{R}^2$.

We will construct G by specifying a finite point set $\mathcal{A} \subset \mathbb{R}^2$ compatible with A and define $G = C_A(\mathcal{A})$. The construction of $\mathcal{A}$ can be divided into several sub-constructions. We start by describing a hexagon of points $\mathcal{H}_k$ for $k \in \mathbb{N}$ which satisfies that $C_A(\mathcal{H}_k)$ is lattice unique.

▶ **Construction 8** ($\mathcal{H}_k$)**.** For an illustration of the construction see the left-hand side of Figure 6. For $x, y \in \mathcal{L}$ we write $d(x, y)$ for the distance between x and y in the graph G_0, and for $k \in \mathbb{N}$ we define $\mathcal{H}_k = \{x \in \mathcal{L} \mid d(x, 0) \in \{k, k+1\}\}$.

Using that G_0 is the infinite triangular grid, it is easy to check that $G_0[\mathcal{H}_k]$ is a lattice unique graph by specifying an enumeration of its vertices satisfying the condition in Definition 7. Moreover, using that e_1 and e_2 satisfy $\|e_1\|_2 = \|e_2\|_2 = \|e_1 - e_2\|_2 = 2$ it follows that the points $\{x \in \mathcal{L} \mid d(x, 0) = k\} \subset \mathcal{H}_k$ lie on a regular hexagon H_k whose corners have Euclidean distance exactly $2k$ to the origin. In particular any point $p \in \mathcal{H}_k$ has $\|p\|_2 \geq \sqrt{3}k$, and thus $\|p\|_A \geq \frac{\sqrt{3}}{2}k$ by Lemma 6.

For a given $\theta \in [0, \pi)$ and $\ell \in \mathbb{N}$ we will construct a set of points $\mathcal{B}_\theta(\ell) \subset \mathbb{R}^2$ compatible with A which constitute a "beam" of argument θ:

▶ **Construction 9** ($\mathcal{B}_\theta(\ell)$)**.** See Figure 6 (right). Let $e_\theta \in \mathbb{R}^2$ be the vector of argument θ with $\|e_\theta\|_A = 2$, and let $f_\theta \in \mathbb{R}^2$ be such that $\|f_\theta\|_A = \|f_\theta - e_\theta\|_A = 2$ (by the URTC property we have two choices for f_θ). For a given $\ell \in \mathbb{N}$ we define

$$\mathcal{B}_\theta(\ell) = \{ae_\theta \mid a \in \{-\ell, \dots, \ell\}\} \cup \{ae_\theta + f_\theta \mid a \in \{-\ell, \dots, \ell - 1\}\}$$

As $\mathcal{B}_\theta(\ell) \subset \mathcal{L}_A(e_\theta, f_\theta)$ it is compatible with A. Moreover it is easy to specify an enumeration of the vertices of $C(\mathcal{B}_\theta(\ell))$ showing that it is lattice unique.

For a given k we want to choose ℓ as large as possible such that $\mathcal{B}_\theta(\ell)$ "fits inside" $G_0[\mathcal{H}_k]$. We then wish to "attach" $\mathcal{B}_\theta(\ell)$ to $G_0[\mathcal{H}_k]$ with extra points $\mathcal{S}$, the number of which does neither depend on k nor on θ. We wish to do it in such a way that $\mathcal{A}_1^k(\theta) := \mathcal{B}_\theta(\ell) \cup G_0[\mathcal{H}_k] \cup \mathcal{S}$ is compatible with A. The precise construction is as follows:

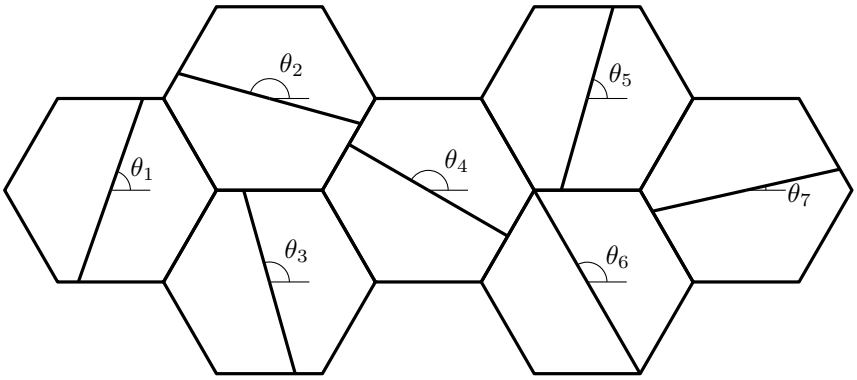


Figure 7 The final point set $\mathcal{A}$ where the point sets $\mathcal{C}_k(\theta)$ are "glued" together by translating them such that the contact graph realized by the union of the subsets $\mathcal{H}_k \subset \mathcal{C}_k(\theta)$ is lattice unique.

▶ **Construction 10** ($\mathcal{C}_k(\theta)$)**.** See Figure 6 (right). Consider the open line segment $L_\theta = \{re_\theta \mid r \in (-r_{\max}, r_{\max})\}$ where $r_{\max}$ is maximal with the property that for all points $x \in L_\theta$ and all $y \in H_k$ it holds that $\|x - y\|_A > 4$. Also let $\ell \in \mathbb{N}$ be maximal such that $\{ae_\theta \mid a \in \{-\ell, \dots, \ell\}\} \subset L_\theta$. Note that

$4 < d_A(\{\ell e_\theta\}, H_k) \leq 6$. Observe moreover that $\ell \geq \frac{\sqrt{3}}{4}k - 3$ as the points $p \in H_k$ have $\|p\|_A \geq \frac{\sqrt{3}}{2}k$. In particular we have the following property which we highlight for later use:

$$\text{If } k > \frac{12}{\sqrt{3}-1} \text{ it holds that } \ell > \frac{k}{4}. \tag{1}$$

When ℓ is chosen in this fashion, we have that $\mathcal{B}_\theta(\ell)$ is contained in the interior of H_k. Now, $\mathcal{B}_\theta(\ell)$ will constitute our beam in direction θ and we will proceed to show that we can attach it to $\mathcal{H}_k$, as illustrated, using only a constant number of extra points. That this can be done is conceptually unsurprising but requires a somewhat technical proof.

We define $\mathcal{S}_1(\theta)$ to be extra points going from the boundary of H_k and z_1^θ to be the extra point which connects $\mathcal{B}_\theta(\ell)$ and $\mathcal{S}_1(\theta)$. This attaches one end of the beam, $\mathcal{B}_\theta(\ell)$, to H_k, and we similarly define $\mathcal{S}_2(\theta)$ and z_2^θ to attach the other end. See Figure 6 (right). In the full version we show that $|\mathcal{S}_i(\theta)| \leq 13$ for $i \in \{1, 2\}$. Letting $\mathcal{C}_k(\theta) = \mathcal{H}_k \cup \mathcal{B}_\theta(\ell) \cup \mathcal{S}_1(\theta) \cup \mathcal{S}_2(\theta) \cup \{z_1^\theta, z_2^\theta\}$ be the combination of the components completes the construction.

We are now ready to construct $\mathcal{A}$ which will consist of several translated copies $\mathcal{C}_k(\theta)$.

▶ **Construction 11** ($\mathcal{A}$)**.** By Lemma 4 we can find an $\varepsilon \in (0, 1)$ and a finite set of directions $\Theta \subset [0, \pi)$ such that for all linear maps $T : \mathbb{R}^2 \to \mathbb{R}^2$ there exists $\theta \in \Theta$ such that

$$\left| \frac{\rho_A(\theta)}{\rho_{T(B)}(\theta)} - 1 \right| \geq \varepsilon. \tag{2}$$

That we can scale the deviation to be multiplicative rather than additive is possible because $0 < \inf_{\theta \in [0,\pi)} \rho_A(\theta) \leq \sup_{\theta \in [0,\pi)} \rho_A(\theta) < \infty$.

For each $\theta \in \Theta$ we construct a copy of $\mathcal{C}_k(\theta) = \mathcal{H}_k \cup \mathcal{B}_\theta(\ell) \cup \mathcal{S}_1(\theta) \cup \mathcal{S}_2(\theta) \cup \{z_1^\theta, z_2^\theta\}$ where k is yet to be fixed (ℓ is of course determined by k and θ). We then choose translations $t_\theta \in \mathbb{R}^2$ for each $\theta \in \Theta$ such that the sets $(\mathcal{H}_k + t_\theta)_{\theta \in \Theta}$ are pairwise disjoint, and $\bigcup_{\theta \in \Theta}(\mathcal{H}_k + t_\theta) \subset \mathbb{R}^2$ is compatible with A and induces a lattice unique contact graph. We can choose $(t_\theta)_{\theta \in \Theta}$ in numerous ways to satisfy this. One is depicted in Figure 7. Another is obtained by

enumerating $\Theta = \{\theta_1, \dots, \theta_q\}$ and defining $t_{\theta_i} = ((2k+3)e_1 - (k+1)e_2) \times (i-1)$. The exact choice is not important and picking one, we define $\mathcal{A}(k) = \bigcup_{\theta\in\Theta}(\mathcal{C}_k(\theta) + t_\theta)$ which is a point set compatible with A. Lastly, we set $\mathcal{A} = \mathcal{A}\left(\lceil \frac{180}{\varepsilon} \rceil\right)$.

We are now ready for the final step of the proof:

Proving that no graph in $C(B)$ contains a subgraph isomorphic to $G = C_A(\mathcal{A})$.

Suppose for contradiction that there exists a set of points $\mathcal{B} \subset \mathbb{R}^2$ such that G is isomorphic to a subgraph of $C_B(\mathcal{B})$. We may clearly assume that $|\mathcal{A}| = |\mathcal{B}|$ and we let $\varphi : \mathcal{A} \to \mathcal{B}$ be a bijection which is also a graph homomorphism when considered as a map $C_A(\mathcal{A}) \to C_B(\mathcal{B})$. The points $\bigcup_{\theta\in\Theta}(\mathcal{H}_k + t_\theta)$ induce a lattice unique contact graph of A. Thus, we may write $\bigcup_{\theta\in\Theta}(\mathcal{H}_k + t_\theta) = \{p_1, \dots, p_n\}$ such that p_1, p_2 and p_3 induce a triangle of G and such that for $i > 3$ there exist distinct $j, k, l < i$ such that p_j, p_k and p_l induce a triangle and such that (p_i, p_k) and (p_i, p_l) are edges of G. By translating the point sets $\mathcal{A}$ and $\mathcal{B}$ we may assume that $\varphi(p_1) = p_1 = 0$. Then applying an appropriate linear transformation T, thus replacing B by $T(B)$, we may assume that $\varphi(p_2) = p_2$ and $\varphi(p_3) = p_3$. Finally, the discussion succeeding Definition 7 implies that in fact $\varphi|_{\bigcup_{\theta\in\Theta}(\mathcal{H}_k+t_\theta)}$ is the identity.

As noted in Construction 11, there exists $\theta \in \Theta$ such that $\left|\frac{\rho_A(\theta)}{\rho_{T(B)}(\theta)} - 1\right| \geq \varepsilon$. The outline of the remaining argument is as follows: The Euclidean distance between the "endpoints" of the beam $\mathcal{B}_\theta(\ell)$ is $2\ell\rho_A(\theta)$, but the rigidity of $\bigcup_{\theta\in\Theta}(\mathcal{H}_k + t_\theta)$ means that it is also $2\ell\rho_{T(B)}(\theta) + O(1)$. When k (and hence ℓ) is large, this will contradict the inequality above.

We refer the reader to the full version of the paper for the technical details. ◀

▶ Remark 12. We claimed that the proof of the part of Theorem 1 concerning unit distance graphs is identical to the proof above. In fact, if we replace $C(X)$ by $U(X)$ for $X \in \{A, B\}$ in the statement of Theorem 2, the result remains valid. To prove it we would construct $\mathcal{A}$ in precisely the same manner. The important point is then that the comments immediately prior to Theorem 1 concerning the rigidity of the realization of lattice unique graphs remains valid. If in particular $\mathcal{B} \subset \mathbb{R}^2$ satisfies that $U_A(\mathcal{A}) \simeq U_A(B)$ via the isomorphism $\varphi : \mathcal{A} \to \mathcal{B}$, we may assume that $\varphi|_{\bigcup_{\theta\in\Theta}(\mathcal{H}_k+t_\theta)}$ is the identity as in the proof above. The remaining part of the argument comparing the lengths of the beams then carries through unchanged. In conclusion, we are only left with the task of proving Theorem 1 for intersection graphs.

3 Intersection Graphs

In this section we prove Theorem 3. Consider two convex bodies A and B. We are going to prove that if $I(A) = I(B)$, then for every graph $G \in C(A)$, there exists a graph $H_k(G) \in I(A)$ with properties as stated in the following lemma.

▶ **Lemma 13.** *Assume that $I(A) = I(B)$. For any $G \in C(A)$ and $k \geq 7$, there exists a graph $H_k(G) \in I(A)$ satisfying the following: Let $X \in \{A, B\}$. For any vertex w of G, there is a corresponding vertex $s_0(w)$ of $H_k(G)$ with the following properties. Consider an arbitrary drawing of $H_k(G)$ as in intersection graph of X and any two vertices w, w' of G and let $s_0 := s_0(w)$ and $s_0' := s_0(w')$. Then $\|s_0 s_0'\|_X \geq 4k - 18$. Furthermore, if ww' is an edge of G, then $\|s_0 s_0'\|_X \leq 4k + 2$.*

As is evident from the lemma, the vertices $(s_0(u))_{u\in V(G)}$, of any drawing of $H_k(G)$ as an intersection graph of X, are placed approximately as the vertices of a drawing of G as a contact graph of scaled convex body $2kX$. To capture the uncertainty, we introduce the concept of *ε-overlap graphs*.

▶ **Definition 14** (ε-overlap Graph). *Let $\varepsilon > 0$ and $A \subset \mathbb{R}^2$ be a symmetric convex body, and let $v_1, \ldots, v_n \in \mathbb{R}^2$ be n points in the plane. Suppose that for any $i, j \in [n]$, $\|v_i v_j\|_A \geq 2 - \varepsilon$. A graph G with vertex set $[n]$ and edge set satisfying $E(G) \subseteq \left\{(i,j) \in [n]^2 \;\middle|\; \|v_i v_j\|_A \leq 2\right\}$ is called an* ε-overlap graph *of A. We say that $\{v_1, \ldots, v_n\}$* realize *the graph G as an ε-overlap graph of A. Further, we denote by $C_\varepsilon(A)$ the set of graphs that can be realized as ε-overlap graphs of A.*

We next show how Lemma 13 leads to a proof of Theorem 3. First, the following lemma provides a reduction from ε-overlap graphs to contact graphs. The proof is a standard compactness argument and can be found in the full version of the paper.

▶ **Lemma 15.** *Let $G_1 = (V, E_1)$ be a graph and A a convex body. If for every $\varepsilon > 0$, it holds that $G_1 \in C_\varepsilon(A)$, then there is a graph $G_2 = (V, E_2) \in C(A)$ such that $E_1 \subseteq E_2$.*

The following lemma uses Lemma 13 to show that if $I(A) = I(B)$, then any $G \in C(A)$ is an ε-overlap graph of B for all $\varepsilon > 0$.

▶ **Lemma 16.** *Assume that $I(A) = I(B)$. For any $G \in C(A)$, and any $\varepsilon > 0$, $G \in C_\varepsilon(B)$.*

Proof. Write $G = (V, E)$ and let $k \geq 7$ be arbitrary. The assumption $I(A) = I(B)$ in particular implies that $H_k(G) \in I(B)$. Consider a drawing of $H_k(G)$ as an intersection graph of B and define $\mathcal{B} := \left\{ \frac{s_0^u}{2k+1} \;\middle|\; u \in V \right\}$. It follows from Lemma 13 that $I_B(\mathcal{B})$ is a drawing of G as a $\left(2 - \frac{4k-18}{2k+1}\right)$-overlap graph of B. Since $\frac{4k-18}{2k+1} \geq 2 - 10/k$, it follows that G is an $10/k$-overlap graph of B. As $k \geq 7$ was arbitrary, the desired result follows. ◀

Theorem 3 is an easy consequence of Lemma 15 and Lemma 16:

Proof of Theorem 3. Let the graph $G \in C(A)$ have the properties of the theorem, i.e., for all $H \in C(B)$, G is not isomorphic to a subgraph of H. Suppose that $I(A) = I(B)$. By Lemma 15 and 16, there is a graph $H = (V, E) \in C(B)$ such that $E' \subseteq E$, which is a contradiction. ◀

It remains to prove Lemma 13. We will proceed to describe the construction of $H_k(G)$ and provide several lemmas needed in order to prove that it satisfies the desired properties.

The proofs of these lemmas and of Lemma 13 are deferred to the full version of the paper.

For each vertex $u \in V(G)$, we make a copy of a graph Q_k to be defined in the following. The vertices of $H_k(G)$ will in turn be the union of the vertices of these copies. We will construct Q_k to have a designated vertex s_0 and a cycle α_k with the property that for every drawing of Q_k as an intersection graph of $X \in \{A, B\}$, the cycle α_k is contained in (and winds all the way around) the annulus $\left\{x \in \mathbb{R}^2 \;\middle|\; \|s_0 x\|_X \in (2k-3, 2k]\right\}$. We may then informally view α_k as an upscaled copy of X up to a slight imprecision that, compared to the size, decreases in k. In order to construct Q_k, we first define another graph P_k (which will be contained in Q_k) with a vertex s_0 such that in every drawing of P_k as an intersection graph, s_0 is contained in k nested disjoint cycles. A priori, it is not clear what it means for s_0 to be contained in a cycle of the graph in every drawing, since the drawing is not necessarily a plane embedding of the graph. However, as the following lemma shows, it is well-defined if P_k is triangle-free. We believe the result to be well-known but have been unable to find the exact formulation that we require in the literature.

▶ **Lemma 17.** *If G is a triangle-free graph then every drawing of G as an intersection graph is a plane embedding.*

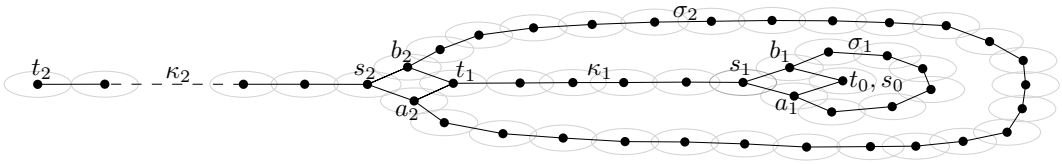


Figure 8 The construction of P_2.

Proof. See the full version of the paper. ◀

We are now ready to define $P_k \in I(A)$ for any $k > 0$. Besides being triangle-free, our aim is that P_k should have the following properties:

1. There is a vertex s_0 such that in any drawing of P_k as an intersection graph of A and B, s_0 is contained in k nested disjoint, simple cycles $\sigma_1, \ldots, \sigma_k$.
2. There is a path κ_k from a vertex s_k to a leaf t_k such that in any drawing of P_k as an intersection graph of A or B, the path κ_k is on the boundary of the outer face.

▶ **Construction 18** (P_k)**.** See Figure 8. We define $P_k = I_A(\mathcal{A}_k)$, where $\mathcal{A}_k$ is a set of points to be defined inductively. Let $\mathcal{A}_0 = \{0\}$ and $P_0 = I_A(\mathcal{A}_0)$ be the trivial graph consisting of one vertex $s_0 = t_0$, which is also the path κ_0. Suppose now that $P_{k-1} = I_A(\mathcal{A}_{k-1})$ has been defined. In order to define P_k, we first add vertices a_k, b_k, s_k and edges such that $\tau_k := (a_k, t_{k-1}, b_k, s_k)$ is a 4-cycle. We now add vertices and edges that together with the path a_k, s_k, b_k form a cycle σ_k. We make σ_k so long that there exists a drawing as an intersection graph in which P_{k-1} is contained in σ_k with respect to both A and B. We finish the construction of P_k by adding vertices and edges that together with s_k form a path κ_k from s_k to a vertex t_k, where κ_k is so long that it cannot be contained in the cycle σ_k, neither as an intersection graph of A nor B. (Note that a path of length n contains $n/2$ independent vertices. A simple volume argument implies a bound on the number of independent vertices contained in the region enclosed by a cycle of a plane intersection embedding.) Let $\mathcal{A}_k$ consist of $\mathcal{A}_{k-1}$ together with all the added points.

▶ **Lemma 19.** *The graph P_k has properties 1–2.*

Proof. See the full version of the paper. ◀

The most important property of P_k is that every vertex $u \in \sigma_k$ has distance $\Omega(k)$ to s_0 in any drawing of P_k as intersection graph of any $X \in \{A, B\}$ in the norm $\|\cdot\|_X$. This is exactly what we will use when constructing Q_k.

▶ **Lemma 20.** *Let $X \in \{A, B\}$. Consider any drawing of P_k as an intersection graph of X. For any vertex $u \in \sigma_k$, we have $\|s_0 u\|_X > 2(k/9 - 1)$.*

Proof. See the full version of the paper. ◀

Having defined P_k we are now ready for the construction of Q_k.

▶ **Construction 21** (Q_k)**.** We here define a graph $Q_k \in I(A)$ by specifying a drawing of Q_k as an intersection graph of A. Let $k' := 18(k+1)$. We start with $P_{k'}$ and explain what to add to obtain Q_k. Let $u_0, \ldots, u_{n-1}$ be the vertices of $\sigma_{k'}$ in cyclic, counter-clockwise order. Consider an arbitrary drawing of $P_{k'}$ as an intersection graph of A and a vertex u_i. Note that $d := \left\lceil \frac{\|s_0 u_i\|_A - 2}{2} \right\rceil$ is the number of vertices needed to add in order to create a path from s_0 to u_i. It follows from Lemma 20 that $d \geq 2k$.

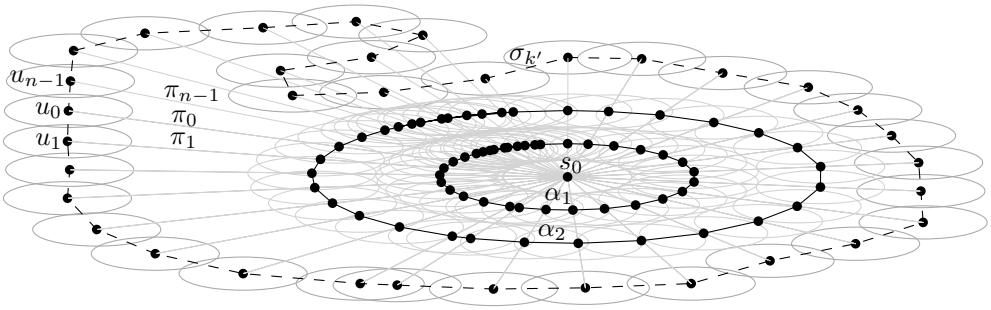


Figure 9 A part of a graph Q_k. The vertices $v_i(j)$ are only shown for $j \in \{1, 2\}$, and only edges on paths π_i and cycles $\alpha_1, \alpha_2, \sigma_{k'}$ are shown.

We want to minimize the vector of these values d with respect to each vertex $u_i \in \sigma_{k'}$. To be precise, we define

$$(d_0, \ldots, d_{n-1}) := \min\left(\left\lceil \frac{\|s_0u_0\|_A - 2}{2} \right\rceil, \ldots, \left\lceil \frac{\|s_0u_{n-1}\|_A - 2}{2} \right\rceil\right),$$

where the minimum is with respect to the lexicographical order and taken over all drawings of $P_{k'}$ as an intersection graph. Consider a drawing of $P_{k'}$ as an intersection graph realizing the minimum and let $\mathcal{P}$ be the set of vertices in the drawing. For each vertex u_i, we create a path π_i from s_0 to u_i as follows. Let $\mathbf{v}_i$ be the unit-vector in direction $u_i - s_0$. We add new vertices placed at the points $v_i(j) := s_0 + 2j\mathbf{v}_i$ for $j \in \{1, \ldots, d_i\}$. We now define the vertices of Q_k as $\mathcal{Q} := \mathcal{P} \cup \bigcup_{i=0}^{n-1}\{v_i(1), \ldots, v_i(d_i)\}$ and define $Q_k = I_A(\mathcal{Q})$. See Figure 9.

▶ Remark 22. By construction, there exists a drawing of Q_k as an intersection graph of A. If there does not exist one of B, we are done, since we then clearly have that $I(A) \neq I(B)$. Now suppose that there exists a drawing of $P_{k'}$ as an intersection graph of B such that

$$\left(\left\lceil \frac{\|s_0u_0\|_B - 2}{2} \right\rceil, \ldots, \left\lceil \frac{\|s_0u_{n-1}\|_B - 2}{2} \right\rceil\right) \prec (d_0, \ldots, d_{n-1}), \tag{3}$$

where $\prec$ denotes the lexicographical order. We can now define a graph $Q_k^B \in I(B)$ from $P_{k'}$ in a similar way as we defined Q_k by adding $\left\lceil \frac{\|s_0u_i\|_B - 2}{2} \right\rceil$ vertices to form a path from s_0 to each u_i. It then follows from (3) that $Q_k^B \notin I(A)$, so in this case we have likewise succeeded in proving $I(A) \neq I(B)$. In the following, we therefore assume that $Q_k \in I(B)$ for any k and that no drawing of $P_{k'}$ as an intersection graph of B satisfying (3) exists.

First we need to show that Q_k contains a cycle α_k as described earlier.

▶ **Lemma 23.** *The set of edges of Q_k contains the pairs $v_i(j)v_{i+1}(j)$ for any $i \in [n]$ and $j \in \{1, \ldots, k\}$, and for each $j \in \{1, \ldots, k\}$, these edges thus form a cycle α_j. In the specific drawing of Q_k as an intersection graph defined in Construction 21, the cycle α_j is contained in the annulus $\left\{x \in \mathbb{R}^2 \mid \|s_0x\|_A \in [2j-1, 2j]\right\}$.*

Proof. See the full version of the paper. ◀

The above lemma shows that the cycle α_k behaves nicely in one particular drawing of Q_k as an intersection graph. To see that something similar holds for every drawing, we refer the reader to the full version.

We now provide the definition of the graph $H_k(G)$, as mentioned in the beginning of this section.

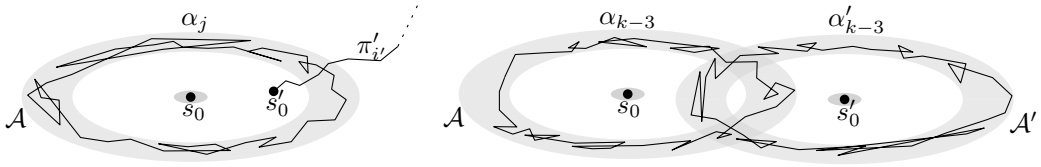


Figure 10 From the proof of Lemma 13 (notation as in Lemma 25). There is either an intersection between α_j and $\pi'_{i'}$ (left) or between α_{k-3} and α'_{k-3} (right) that violates Lemma 25.

▶ **Construction 24** ($H_k(G)$)**.** For any $G \in C(A)$, consider a fixed drawing of G as a contact graph of A. For each vertex w of G, we make a copy of the drawing of Q_k as an intersection graph as defined in Construction 21 which we translate so that s_0 is placed at $s_0^w := (2k-2)w$. We then add all edges induced by the vertices, and the result is denoted as $H_k(G)$.

The following lemma characterizes some of the edges of $H_k(G)$ and will be crucial in the proof of Lemma 13.

▶ **Lemma 25.** *Consider two vertices w, w' of a drawing of a graph G as a contact graph. Denote by Q and Q' the copies of Q_k in $H_k(G)$ corresponding to w and w', respectively, such that $s_0, \pi_i, \alpha_j, v_i(j)$ denote objects in Q and $s'_0, \pi'_i, \alpha'_j, v'_i(j)$ denote objects in Q'. If $v_i(j)v'_{i'}(j')$ is an edge of $H_k(G)$, then $j + j' \geq 2k - 4$.*

If ww' is an edge of G, then there is an edge $v_i(k)v'_{i'}(k)$ in $H_k(G)$.

Proof. See the full version of the paper. ◀

As mentioned, the final proof of Lemma 13 is deferred to the full version, but we can now provide the main ideas. We first need to prove that in any drawing of Q_k as an intersection graph with respect to $X \in \{A, B\}$, any cycle α_j is contained in an annulus only slightly wider than as stated in Lemma 23. Furthermore, α_j winds around s_0 in the sense that if we trace the full curve α_j, the change of argument with respect to s_0 will be $\pm 2\pi$. To prove the lower bound $\|s_0 s'_0\|_X \geq 4k - 18$ in Lemma 13, we exclude that the distance is smaller by dividing into two cases depending on the actual distance. Figure 10 depicts the two cases for each of which we prove that there would be an edge in $H_k(G)$ violating Lemma 25. The upper bound $\|s_0 s'_0\|_X \leq 4k + 2$ when ww' is an edge of G is likewise an easy consequence of Lemma 25, as otherwise, an edge $v_i(k)v'_{i'}(k)$ would be missing from $H_k(G)$.

4 Concluding remarks

It is natural to investigate the special case of convex bodies with the URTC property. Here our proof of Theorem 2 fails since the hexagons are not rigid structures. Together with Konrad Swanepoel, we have promising progress in generalizing Theorem 1 to also handle this case.

Another interesting direction is to consider convex bodies in three and higher dimensions. Already in three dimensions, it appears to be very difficult to characterize when two bodies induce the same graph classes.

References

1 Anders Aamand, Mikkel Abrahamsen, Jakob Bæk Tejs Knudsen, and Peter Michael Reichstein Rasmussen. Classifying convex bodies by their contact and intersection graphs, 2019. Preprint. arXiv:1902.01732.

2 Édouard Bonnet, Nicolas Grelier, and Tillmann Miltzow. Maximum clique in disk-like intersection graphs. *Preprint*, 2020. arXiv:2003.02583.

3 Károly Böröczky Jr. *Finite packing and covering*, volume 154 of *Cambridge Tracts in Mathematics*. Cambridge University Press, 2004.

4 Sergio Cabello and Miha Jejčič. Refining the hierarchies of classes of geometric intersection graphs. *The Electronic Journal of Combinatorics*, 24(1):1–19, 2017.

5 Jean Cardinal. Computational geometry column 62. *SIGACT News*, 46(4):69–78, 2015.

6 Jean Cardinal, Stefan Felsner, Tillmann Miltzow, Casey Tompkins, and Birgit Vogtenhuber. Intersection graphs of rays and grounded segments. In Hans L. Bodlaender and Gerhard J. Woeginger, editors, *Graph-Theoretic Concepts in Computer Science*, pages 153–166, Cham, 2017. Springer International Publishing.

7 Timothy M. Chan and Dimitrios Skrepetos. Approximate shortest paths and distance oracles in weighted unit-disk graphs. *Journal of Computational Geometry*, 10(2), 2019.

8 Steven Chaplick, Stefan Felsner, Udo Hoffmann, and Veit Wiechert. Grid intersection graphs and order dimension. *Order*, 35:363–391, 2018.

9 Brent N. Clark, Charles J. Colbourn, and David S. Johnson. Unit disk graphs. *Discrete Mathematics*, 86(1-3):165–177, 1990.

10 László Fejes Tóth. *Lagerungen in der Ebene auf der Kugel und im Raum*, volume 65 of *Die Grundlehren der mathematischen Wissenschaften*. Springer, second edition, 1972.

11 Stefan Felsner and Günter Rote. On primal-dual circle representations. In *34th European Workshop on Computational Geometry (EuroCG 2018)*, pages 72:1–72:6, 2018.

12 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, and Meirav Zehavi. Finding, hitting and packing cycles in subexponential time on unit disk graphs. *Discrete & Computational Geometry*, 62(4):879–911, 2019.

13 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, and Meirav Zehavi. ETH-Tight Algorithms for Long Path and Cycle on Unit Disk Graphs. In *36th International Symposium on Computational Geometry (SoCG 2020)*, pages 44:1–44:18, 2020. doi:10.4230/LIPIcs.SoCG.2020.44.

14 Stefan Funke, Alexander Kesselman, Ulrich Meyer, and Michael Segal. A simple improved distributed algorithm for minimum CDS in unit disk graphs. *ACM Transactions on Sensor Networks*, 2(3):444–453, 2006. doi:10.1145/1167935.1167941.

15 György Pál Gehér. A contribution to the Aleksandrov conservative distance problem in two dimensions. *Linear Algebra and its Applications*, 481:280–287, 2015.

16 Albert Gräf, Martin Stumpf, and Gerhard Weißenfels. On coloring unit disk graphs. *Algorithmica*, 20(3):277–293, 1998.

17 Svante Janson and Jan Kratochvíl. Thresholds for classes of intersection graphs. *Discrete Mathematics*, 108:307–326, 1992.

18 Victor Klee. Some new results on smoothness and rotundity in normed linear spaces. *Mathematische Annalen*, 139(1):51–63, 1959.

19 P. Koebe. Kontaktprobleme der konformen Abbildung. *Berichte über die Verhandlungen der Sächsische Akademie der Wissenschaften zu Leipzig, Mathematisch–Physische Klasse*, 88:141–164, 1936.

20 Jan Kratochivíl and Jiří Matoušek. Intersection graphs of segments. *Journal of Combinatorial Theory Series B*, 62(2):289–315, 1994.

21 M. V. Marathe, H. Breu, H. B. Hunt III, S. S. Ravi, and D. J. Rosenkrantz. Simple heuristics for unit disk graphs. *Networks*, 25(2):59–68, 1995. doi:10.1002/net.3230250205.

22 Colin McDiarmid and Tobias Müller. Integer realizations of disk and segment graphs. *Journal of Combinatorial Theory, Series B*, 103(1):114–143, 2013.

23 Tobias Müller, Erik Jan van Leeuwen, and Jan van Leeuwen. Integer representations of convex polygon intersection graphs. *SIAM Journal on Discrete Mathematics*, 27(1):205–231, 2013.

24 Fahad Panolan, Saket Saurabh, and Meirav Zehavi. Contraction decomposition in unit disk graphs and algorithmic applications in parameterized complexity. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2019)*, pages 1035–1054, 2019.

25 Marco A. Peyrot-Solís, Giselle M. Galvan-Tejada, and Hildeberto Jardon-Aguilar. Proposal of a planar directional UWB antenna for any desired operational bandwidth. *International Journal of Antennas and Propagation*, 2014:1–12, 2014.

26 Oded Schramm. Combinatorically prescribed packings and applications to conformal and quasiconformal maps. *Preprint*, 2007. `arXiv:0709.0710`.

27 Konrad Swanepoel. Combinatorial distance geometry in normed spaces. In Gergely Ambrus, Imre Bárány, Károly J. Böröczky, Gábor Fejes Tóth, and János Pach, editors, *New Trends in Intuitive Geometry*, volume 27 of *Bolyai Society Mathematical Studies*. Springer, 2018.

28 G. Fejes Tóth. *New Results in the Theory of Packing and Covering*, pages 318–359. Birkhäuser Basel, Basel, 1983. `doi:10.1007/978-3-0348-5858-8_14`.

29 Weili Wu, Hongwei Du, Xiaohua Jia, Yingshu Li, and Scott C-H Huang. Minimum connected dominating sets and maximal independent sets in unit disk graphs. *Theoretical Computer Science*, 352(1-3):1–7, 2006.

30 Tudor Zamfirescu. Nearly all convex bodies are smooth and strictly convex. *Monatshefte für Mathematik*, 103(1):57–62, 1987.

Approximate Nearest-Neighbor Search for Line Segments

Ahmed Abdelkader
Oden Institute for Computational Engineering and Sciences,
The University of Texas at Austin, TX, USA

David M. Mount
Department of Computer Science and Institute of Advanced Computer Studies,
University of Maryland, College Park, MD, USA

Abstract

Approximate nearest-neighbor search is a fundamental algorithmic problem that continues to inspire study due its essential role in numerous contexts. In contrast to most prior work, which has focused on point sets, we consider nearest-neighbor queries against a set of line segments in $\mathbb{R}^d$, for constant dimension d. Given a set S of n disjoint line segments in $\mathbb{R}^d$ and an error parameter $\varepsilon > 0$, the objective is to build a data structure such that for any query point q, it is possible to return a line segment whose Euclidean distance from q is at most $(1 + \varepsilon)$ times the distance from q to its nearest line segment. We present a data structure for this problem with storage $O((n^2/\varepsilon^d)\log(\Delta/\varepsilon))$ and query time $O(\log(\max(n, \Delta)/\varepsilon))$, where Δ is the spread of the set of segments S. Our approach is based on a covering of space by anisotropic elements, which align themselves according to the orientations of nearby segments.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Approximate nearest-neighbor searching, Approximate Voronoi diagrams, Line segments, Macbeath regions

Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.4

Related Version *Full Version*: `https://arxiv.org/abs/2103.16071`

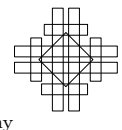
Funding Supported by NSF grant CCF-1618866 and an Ann G. Wylie Dissertation Fellowship.

Acknowledgements The first author's work was conducted in part at the University of Maryland.

1 Introduction

Proximity queries are essential building blocks in many important algorithms with numerous applications [21–25, 28–30]. A primary example is *nearest-neighbor searching*, where a given set of n points P in $\mathbb{R}^d$ is preprocessed into a data structure so that queries can be answered efficiently. As the complexity bounds for such query problems grow very rapidly as the dimension increases, either in terms of query time or space, most research has focused on approximate solutions. There has been a great deal of work on approximate proximity searching in spaces of very high dimension [5, 19, 32, 34] and in general metric spaces [5, 33, 39, 40]. Nonetheless, there are many important applications that naturally reside in real spaces of relatively low dimensions.

In this paper, we consider approximate nearest-neighbor searching for a query point against a discrete set of line segments in $\mathbb{R}^d$, where d is a fixed constant. We are given a set S of n disjoint line segments in $\mathbb{R}^d$. The distance from any point $q \in \mathbb{R}^d$ to a segment s, denoted $\text{dist}(q, s)$, is the minimum Euclidean distance between q and any point of s. For $\varepsilon > 0$, a segment $s' \in S$ is an *ε-approximate nearest neighbor* (*ε-ANN*) of q if $\text{dist}(q, s')$ is within a factor of $1 + \varepsilon$ of the distance to q's closest segment in S. Given S and $\varepsilon > 0$, the objective is to construct a data structure so that given any $q \in \mathbb{R}^d$, it is possible to compute an ε-ANN of q efficiently. We refer to this problem as *segment-ANN*.



Clearly, nearest-neighbor searching with respect to segments is at least as hard as for point sets, and there are quadratic worst-case lower bounds in the exact and approximate settings [6,31]. The difficulty of the problem can be appreciated by considering the increased complexity of the Voronoi diagram of a set of lines or line segments, which is not fully understood to date [14,16]. Recall that such Voronoi diagrams consist of cells bounded by hyperplanes and algebraic surfaces of constant degree, and hence are generally nonconvex; see [36,49] for the computation of Voronoi diagrams of line segments in $\mathbb{R}^2$. A set of just three straight lines in $\mathbb{R}^3$ suffices to induce a highly intricate Voronoi diagram [27]. Better bounds are known in restricted scenarios, for example, by bounding the number of orientations [26,37] or working with a polyhedral distance function [20,38].

Our work follows in this tradition with the main motivation of developing a better understanding of proximity searching among more complex objects than discrete sets of points. We are particularly interested in distance functions whose rate of change is much larger in some directions compared to others. This sort of behavior is characterized by the notion of *anisotropy*, which can be defined for smooth convex functions as the ratio of the largest to the smallest eigenvalues of the Hessian matrix at the point in question. Line segments are perhaps the simplest objects inducing such distance functions. This type of proximity searching against linear and affine subspaces has recently been applied to problems in pattern recognition [17,51] and active learning [50]. A notion of *direction-sensitive* distances in the plane has been studied in [3].

In this paper we present a new data structure for segment-ANN. Our data structure is an AVD-style data structure [9,12,13,31]. By this we mean that it employs a hierarchical subdivision of space (a covering in our case) by elements of constant complexity (ellipsoids in our case). At the leaf level of the hierarchy, each element stores a *representative segment* of S that is an ε-ANN for any query point lying within the element. Queries are answered by a simple descent through the hierarchy, reporting the representative of the leaf-level element. Up to now, AVD structures have relied on quadtree-based subdivisions. A novel feature of our approach is that the elements are anisotropic, where their shapes are sensitive to the local distribution of segments. The advantages of such an approach are illustrated intuitively in Figure 1. Our approach is inspired by recent progress on the use of anisotropic covering elements based on Macbeath regions [15,42] used in convex approximation [1,7,8,10,11].

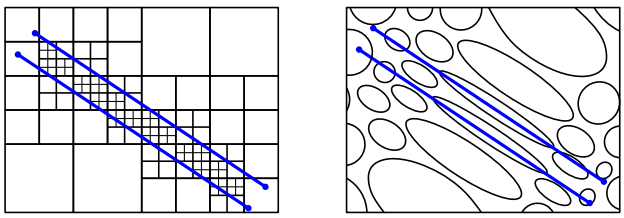


Figure 1 Approximation using isotropic (quadtree) elements compared to anisotropic elements.

Our input consists of a set S of n pairwise disjoint line segments in $\mathbb{R}^d$. Define the *spread*, denoted $\Delta(S)$ to be $\mathrm{diam}(S)/\delta_{\min}(S)$, where $\mathrm{diam}(S)$ is the diameter of the set S (the maximum distance between any two points lying on these segments), and $\delta_{\min}$ is the minimum distance between any two segments. Since S will be fixed throughout, we will just refer to this as Δ. Here is our main result.

▶ **Theorem 1.** *Given a set S of n disjoint line segments in $\mathbb{R}^d$ of spread Δ and $\varepsilon > 0$, there exists a data structure that can answer ε-ANN queries in time $O(\log(\max(n,\Delta)/\varepsilon))$ using $O((n^2/\varepsilon^d)\log\frac{\Delta}{\varepsilon})$ storage.*

Note that because the line segments are disjoint and the dimension is constant, $n = O(\Delta^d)$, the query time bound can be simplified to $O(\log \frac{\Delta}{\varepsilon})$.

The most closely related works to ours are segment-ANN data structures by Mahabadi [43] and Agarwal, Rubin, and Sharir [2]. Mahabadi's solution is based on reducing segment-ANN to point-ANN through a combination of reductions. These reductions produce $n^{O(1)}$ point-ANN modules, where each module involves $O(n/\varepsilon^{O(1)})$ points. The space bounds obtained are inferior to ours in terms of n and ε, and while individual modules can be solved within the AVD model, the overall data structure is not in this model. Agarwal, Rubin, and Sharir [2] consider the more general problem of ANN queries against k-flats in $\mathbb{R}^d$. As in our case, d is assumed to be constant. They solve the problem by approximating the Euclidean ball with a polyhedron distance function of complexity $1/\varepsilon^{O(1)}$ [18], and they show that it is possible to compute nearest neighbors exactly among k-flats with respect to the induced polyhedral distance function through the use of multi-level partition trees. In the case of line segments, their approach provides polylogarithmic query time with $n^2(\log(n)/\varepsilon)^{O(1)}$ storage, but the approach makes critical use of the fact that the objects are (infinite) flats. As with Mahabadi's result, there is no dependence on the spread. There are also works that consider the problem in its dual form, where the data set consists of points and the query is a k-flat [2, 4, 44].

Our data structure has a number of notable features. First, it is in the AVD model (which partially answers an open problem posed by Agarwal, Rubin, and Sharir [2]. The query algorithm is almost trivial, involving a descent through a rooted directed acyclic graph (DAG) of constant degree. The decision of which neighbor to visit next is just a membership test for an ellipsoid. By abandoning the quadtree-based approaches used in prior AVD solutions, we demonstrate how to exploit the anisotropic nature of the nearest-neighbor distance function to obtain a space-efficient hierarchical spatial decomposition.

The remainder of the paper is organized as follows. Section 2 formalizes the notion of anisotropy by examining the differential properties of the distance to the segments. Section 3 introduces the notion of a *capsule*, the basic shape upon which our data structure is built and introduces the relevant properties of these objects. Section 4 presents our ANN search structure, and finally Section 5 analyzes its storage requirements.

2 Exposing Anisotropy

In this section, we formally characterize how the distance function associated by a set of line segments naturally induces a Riemannian metric whose metric tensor is anisotropic; see Figure 2. This characterization underpins the design of our data structure which draw inspiration from classical constructions in convex optimization. For the sake of efficiency, the construction of our data structure will be based on a simpler approach, and this section may be skimmed without hampering the understanding of the material that follows.

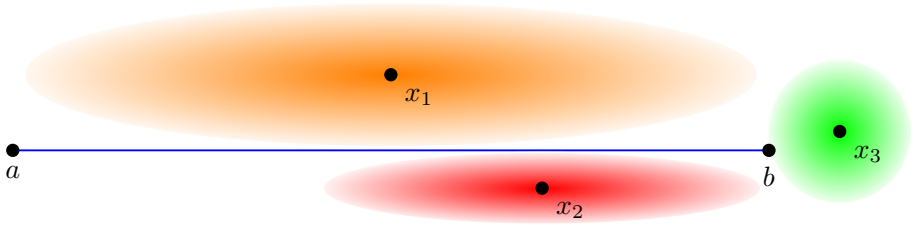


Figure 2 Demonstrating the local tensors induced by a segment $\overline{ab}$ as defined in Equation 5.

Given a set of pairwise-disjoint segments $S = \{s_1, \ldots, s_n\}$, denote by ℓ_i the line supporting $s_i = \overline{a_i b_i}$ parallel to the unit vector v_i. We define the distance functions at any $x \in \mathbb{R}^d$ as

$$D_i(x) \;=\; \begin{cases} D_i^{\ell}(x), & \text{if } x^{\perp} \in \operatorname{int}(s_i), \\ D_i^{\bullet}(x), & \text{otherwise,} \end{cases} \tag{1}$$

where D_i^{ℓ} is half the squared distance to the line ℓ_i, and $D_i^{\bullet}(x) = \min\{D^{a_i}(x), D^{b_i}(x)\}$ with D^{a_i} and D^{b_i} being half the squared distance to the endpoints a_i and b_i, respectively, $x^{\perp}$ the projection of x onto ℓ_i, and $\operatorname{int}(s_i)$ the interior of s_i. As is common for similar definitions, we work with squared distances and introduce the $\frac{1}{2}$ factor to simplify the resulting derivatives.

For every $x \in \mathbb{R}^d$, we seek a definition of a *local tensor* to effectively consolidate the two cases in the definition of $D_i(x)$ per Equation 1. Using such local tensors, we can define a *local descriptor*, e.g., an ellipsoid, whose shape describes the rate of change of the distance function D_i in the neighborhood of x. We achieve this by first examining the Hessian of the distance functions defining $D_i(x)$ for each segment in isolation. Then, we consider the consolidation of all distance functions as needed for nearest-neighbor searching.

2.1 Distance Hessians

For a fixed point $p \in \mathbb{R}^d$, the associated distance takes the form

$$D^p(x) \;=\; \frac{1}{2}\|x - p\|^2 \;=\; \frac{1}{2}\sum_{i=1}^{d}(x_i - p_i)^2, \quad \text{for which } \nabla^2 D^p = I, \tag{2}$$

where ∇^2 denotes the function's Hessian and I is the identity matrix. For a fixed line $\ell = \{p + tv \mid t \in \mathbb{R}\}$, with $p, v \in \mathbb{R}^d$ and $\|v\| = 1$, the distance takes the form

$$D^{\ell}(x) \;=\; \frac{1}{2}\|x - x^{\perp}\|^2 \;=\; \frac{1}{2}\sum_{i=1}^{d}\left((x_i - p_i) - \langle x - p, v\rangle v_i\right)^2,$$

where $x^{\perp}$ is the projection of x onto ℓ. We proceed to compute the Hessian $\nabla^2 D^{\ell}$ as follows.

$$\frac{\partial D^{\ell}}{\partial x_k} \;=\; (x_k - p_k) - \langle x - p, v\rangle v_k, \qquad \frac{\partial^2 D^{\ell}}{\partial x_k^2} \;=\; 1 - v_k^2, \qquad \frac{\partial^2 D^{\ell}}{\partial x_k \partial x_n} \;=\; -v_k v_n,$$

$$\nabla^2 D^{\ell} \;=\; I - vv^{\intercal}. \tag{3}$$

It is easy to verify that v is an eigenvector of $\nabla^2 D^{\ell}$ with eigenvalue 0. Letting T be any rotation matrix such that $Tv = [1, 0, \ldots, 0]^{\intercal}$, we obtain

$$\nabla^2(D^{\ell} \circ T) \;=\; \begin{bmatrix} 0 & 0 & \ldots & 0 \\ 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 1 \end{bmatrix}.$$

Noting that $\nabla^2 D^{\ell} = T^{-1}\nabla^2(D^{\ell} \circ T)T^{-1}$, and that eigenvalues are invariant under change of basis, the remaining eigenvalues of $\nabla^2 D^{\ell}$ are all equal to 1, where the corresponding eigenvectors can be chosen as any basis of the subspace orthogonal to v. This form of the Hessian reflects the constancy of the distance along trajectories parallel to the line.

2.2 Local Tensors and Ellipsoids

Per the previous subsection, the Hessian $\nabla^2 D_i^\ell$, of the distance to a line ℓ_i, is rank-deficient with v_i an eigenvector with eigenvalue 0 and all remaining eigenvalues equal to 1. We remedy this deficiency by defining the local tensor as

$$\mathbb{H}_i(x) \;=\; \frac{1}{D_i(x)}\nabla^2 D_i^\ell + \frac{1}{D_i^\bullet(x)} v_i v_i^\intercal. \tag{4}$$

By construction, the local tensor $\mathbb{H}_i(x)$ has a single eigenvalue equal to $1/D_i^\bullet(x)$ with all remaining $d-1$ eigenvalues equal to $1/D_i(x)$. The anisotropy of the distance function D_i reflected by this local tensor at x is equal to the ratio of the maximum of $D_i^\bullet(x)$ and $D_i(x)$ to their minimum. As x moves along any smooth trajectory, this anisotropy varies continuously between of 1 and ∞. We use the tensor $\mathbb{H}_i(x)$ to define the ellipsoid

$$\mathcal{E}_i(x) \;=\; \left\{ y \in \mathbb{R}^d \;\middle|\; \frac{1}{2}(y-x)^\intercal \mathbb{H}_i(x)(y-x) \le 1 \right\}. \tag{5}$$

(See Figure 2 for examples.) Observe that as $D_i^\bullet$ becomes larger, the eigenvalue associated with v_i becomes smaller, and the ellipsoid $\mathcal{E}_i(x)$ extends further in the direction of v_i. On the other hand, as $D_i^\ell(x)$ approaches $D_i^\bullet(x)$, $\mathbb{H}_i(x)$ approaches a scaled identity matrix, and the ellipsoid $\mathcal{E}_i(x)$ becomes more spherical.

One approach to account for the influence of all n segments is to define a *blended tensor* at every $x \in \mathbb{R}^d$, $\mathbb{H}(x) = \sum_{i=1}^n \mathbb{H}_i(x)$ along with an induced *local norm*[1] and a corresponding *local ellipsoid* acting as a *metric ball* at x

$$\widetilde{\mathcal{E}}(x) \;=\; \left\{ y \in \mathbb{R}^d \mid \|y-x\|_x^2 \le 1 \right\}, \quad \text{where} \quad \|y-x\|_x^2 \;=\; \frac{1}{2}(y-x)^\intercal \mathbb{H}(x)(y-x). \tag{6}$$

Alternatively, we may directly bound the relative change of *all* distance functions in the neighborhood of x by restricting attention to the cell [2]

$$\widehat{\mathcal{E}}(x) \;=\; \bigcap_{i=1}^n \mathcal{E}_i(x). \tag{7}$$

The next lemma formalizes the relationship between the ellipsoids $\widetilde{\mathcal{E}}(x)$ and the cells $\widehat{\mathcal{E}}(x)$.[3]

▶ **Lemma 2.** *For any set of n segments and any point $x \in \mathbb{R}^d$, we have the inclusions*

$$\widetilde{\mathcal{E}}(x) \;\subseteq\; \widehat{\mathcal{E}}(x) \;\subseteq\; \widetilde{\mathcal{E}}^{\sqrt{n}}(x), \quad \textit{where the superscript denotes the central scaling about } x.$$

Proof. The first inclusion is immediate. For the second inclusion, observe that any $y \in \widehat{\mathcal{E}}(x)$ satisfies $\max_i \frac{1}{2}(y-x)^\intercal \mathbb{H}_i(x)(y-x) \le 1$. Hence, $\|y-x\|_x^2 \le n$, implying $y \in \widetilde{\mathcal{E}}^{\sqrt{n}}(x)$. ◀

Unfortunately, each of those two approaches has its own drawbacks. While the blended tensors $\mathbb{H}(x)$ are easier to compute, the corresponding ellipsoids $\widetilde{\mathcal{E}}(x)$ are unnecessarily small. On the other hand, the cells $\widehat{\mathcal{E}}(x)$ can retain a suitable size but are difficult to construct. This motivates an alternative, and more geometric, definition of a more efficient shape primitive.

---

[1] See [45, 47] for related derivations of Riemannian metrics from local tensors.

[2] This can be seen by recognizing $\mathbb{H}_i(x)$ as the Hessian of a closely related function derived from D_i, and writing its Taylor expansion about x for points within $\widetilde{\mathcal{E}}(x)$.

[3] Readers familiar with the *Dikin ellipsoid* from convex optimization will recognize the similarities with the local ellipsoids $\widetilde{\mathcal{E}}(x)$. It is well-known that Macbeath regions and Dikin ellipsoids are related by a similar inclusion as in Lemma 2: for a polytope K defined as the intersection of m halfspaces and a point $x \in K$, the Macbeath region $K \cap (2x - K)$ contains the Dikin ellipsoid at x and is contained in its $\sqrt{m}$ expansion; see, e.g., [41, 48]. While Dikin ellipsoids are derived from barrier functions [46, 52], we derive our ellipsoids from the Euclidean distance functions.

3 Anisotropic Space Covers

Building upon the derivations in the previous section, we propose a simple primitive shape for constructing a hierarchical space covering, which will be amenable to computation and analysis. Recall that S is a set of n disjoint line segments in $\mathbb{R}^d$, each defined by its two endpoints. Fix a segment $s = \overline{ab} \in S$, and let $r > 0$ be a given distance parameter, to be defined later. Recall that for any $x \in \mathbb{R}^d$, its distance to segment s is denoted by $\mathrm{dist}(x, s)$.

For any point $x \in \mathbb{R}^d$, we define a convex and centrally-symmetric subregion centered about x, called a *capsule* and denote by $\mathcal{C}_s(x, r)$. If the closest point to x on s is an endpoint, then $\mathcal{C}_s(x, r)$ is simply the ball centered at x with radius $\max(r, \mathrm{dist}(x, s))$. Otherwise, $\mathcal{C}_s(x, r)$ is defined as follows. First, construct the infinite cylinder of radius $\max(r, \mathrm{dist}(x, s))$ with axis parallel to s and passing through x. Consider a ball centered at x whose radius is $\max(r, \min(\|x - a\|, \|x - b\|)$, where a and b are the endpoints of s. The capsule is the intersection of this cylinder and ball (see Figure 3(a) and (b)).

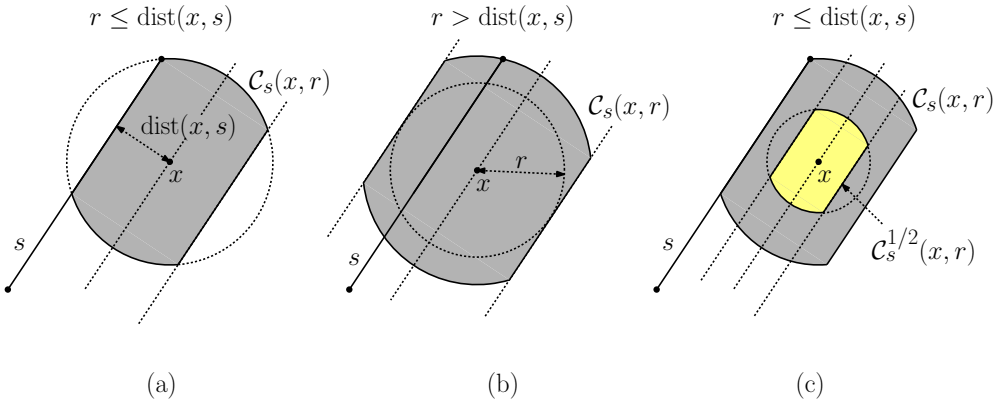


Figure 3 (a) The capsule $\mathcal{C}_s(x, r)$ at x for segment s for $r \le \mathrm{dist}(x, s)$, (b) for $r > \mathrm{dist}(x, s)$, and (c) the shrunken capsule $\mathcal{C}_s^{1/2}(x, r)$ for $r \le \mathrm{dist}(x, s)$.

We define the *capsule* associated with x for the set S of all segments as $\mathcal{C}_S(x, r) = \bigcap_{s \in S} \mathcal{C}_s(x, r)$. Since S will be fixed throughout, we will omit this subscript henceforth. Clearly, $\mathcal{C}(x, r)$ is also convex and centrally symmetric about x. We start by showing that capsules are closely related to the Hessian-based ellipsoids defined in Eq. (5).

▶ **Lemma 3.** *For all $x \in \mathbb{R}^d$ and any $r \le \min_i \mathrm{dist}(x, s_i)$,*

$$\widehat{\mathcal{E}}(x) \;\subseteq\; \mathcal{C}(x, r) \;\subseteq\; \widehat{\mathcal{E}}^{\sqrt{2}}(x). \tag{8}$$

Proof. By the definition of the local tensor per Equation 4, we may express the capsule as

$$\mathcal{C}_{s_i}(x, r) \;=\; \left\{ y \in \mathbb{R}^d \;\middle|\; \max_i \left(\frac{(y-x)^\intercal \nabla^2 D_i^\ell (y-x)}{\max\{r^2, 2 \cdot D_i(x)\}}, \frac{(y-x)^\intercal (y-x)}{\max\{r^2, 2 \cdot D_i^\bullet(x)\}} \right) \le 1 \right\}, \tag{9}$$

where, in contrast to Equation 4, we replaced $v_i v_i^\intercal$ in the second term by just the identity matrix to make the shape nicer. Recognizing the definition of both $\widehat{\mathcal{E}}$ and the capsule $\mathcal{C}$ as the intersection of n subsets, it suffices to establish the following claim: for all segments s_i, and any $r \le \mathrm{dist}(x, s_i)$, we have that

$$\mathcal{E}_i(x) \;\subseteq\; \mathcal{C}_{s_i}(x, r) \;\subseteq\; \mathcal{E}_i^{\sqrt{2}}(x). \tag{10}$$

Observing that $D_i(x) = \text{dist}^2(x, s_i)/2$ and $D_i(x) \leq D_i^\bullet(x)$, we see that r^2 cannot dominate in either of the denominators in Equation 9. In addition, for any $y \in \mathbb{R}^d$ we may write $y - x = \alpha v + \beta u$, where u is a unit vector orthogonal to v. We obtain

$$y \in \mathcal{E}_i(x) \implies \frac{1}{2}(y-x)^\intercal \mathbb{H}_i(x)(y-x) = \frac{1}{2}\left(\frac{\beta^2}{D_i(x)} + \frac{\alpha^2}{D_i^\bullet(x)}\right) \leq 1.$$

From the above, the first term in Equation 9 is at most 1. For the second term, observe that $\beta^2/D_i^\bullet(x) \leq \beta^2/D_i(x)$. By making this substitution, we find that $\frac{1}{2}(\beta^2 + \alpha^2)/D_i^\bullet(x) = \frac{1}{2}\|y-x\|^2/D_i^\bullet(x) \leq 1$, implying the second term in Equation 9 is at most 1 as well. It follows that $y \in \mathcal{C}_{s_i}(x, r)$, establishing the first inclusion. For the second inclusion, observe that for all $y \in \mathcal{C}_{s_i}(x, r)$ we have that $\frac{1}{2}(y-x)^\intercal \mathbb{H}_i(q)(y-x) \leq 2$. This holds as both terms in Equation 9 are at most 1, and we have $(y-x)^\intercal vv^\intercal(y-x) \leq (y-x)^\intercal(y-x)$ for all unit vectors v. Using Equation 10, the proof follows by intersection over all $i \in [n]$. ◀

For the purposes of distance approximation, we work with a scaled version of these capsules which we denote by the superscript $\mathcal{C}^\lambda$ for a scale factor λ. The scaled version of each region is the central scaling around x by λ. When $\lambda < 1$, we say that the regions are *shrunken* (see Figure 3(c)).

Capsules enjoy a number of useful properties, similar to the Macbeath regions in the context of convex bodies; see [1, 15]. In particular, capsules satisfy the following *expansion-containment property*, which states that whenever two shrunken capsules overlap, a constant factor expansion of one contains the other.

▶ **Lemma 4.** *Let S be a set of disjoint line segments and $\lambda \in (0, 1)$ be a scale factor. For any $x, y \in \mathbb{R}^d$, if $\mathcal{C}^\lambda(x, r) \cap \mathcal{C}^\lambda(y, r) \neq \emptyset$ then $\mathcal{C}^\lambda(y, r) \subseteq \mathcal{C}^{\alpha\lambda}(x, r)$, where $\alpha = \frac{3+\lambda}{1-\lambda}$.*

3.1 Local Feature Size

In order to use capsules for space covering, we need a principled way to select the distance parameter r. We define the *local feature size* (*LFS*) at $x \in \mathbb{R}^d$ as the distance from x to the *second-nearest* segment:

$$\phi(x) = \min_{i,j \in \binom{n}{2}} \max\{\text{dist}(x, s_i), \text{dist}(x, s_j)\}, \tag{11}$$

where we assume $n \geq 2$. It is easy to see that ϕ is 1-Lipschitz, and the following lemma further quantifies the sensitivity of capsules to the distances to the set of line segments. In particular, all points within a shrunken capsule have comparable local feature size.

▶ **Lemma 5.** *For all $z \in \mathcal{C}^\lambda(x, \phi(x))$, where $0 < \lambda < 1$, $\phi(z) \in [1-\lambda, 1+\lambda] \cdot \phi(x)$.*

Proof. Let $r = \phi(x)$, and denote by s_1 and s_2 the nearest and second-nearest segments to x, respectively. Observing that $r = \text{dist}(x, s_2)$, the interior of $\mathcal{C}(x, r)$ cannot intersect any segment except for s_1. To obtain the lower bound, we bound the distance from any $z \in \mathcal{C}^\lambda(x, r)$ to $\partial\mathcal{C}(x, r)$.

Recalling the construction of capsules, for any segment s, as an infinite cylinder restricted within a ball, we define the *spine* of the capsule at x with respect to s as the projection of s onto the axis of the cylinder intersected with the capsule, or just x if the capsule is a ball; see Figure 4.

For any $z \in \mathcal{C}_{s_2}(x, r)$, define z' as the projection of z onto the spine. By construction, $\text{dist}(z, z') \leq r$ and $\text{dist}(z', s_2) = r$, for all $z \in \mathcal{C}_{s_2}(x, r)$ and z' on the spine. Upon shrinking, we obtain $\text{dist}(z, z') \leq \lambda r$. In addition, $\text{dist}(z, \partial\mathcal{C}(x, r)) \geq (1-\lambda)r$. This lower bound is

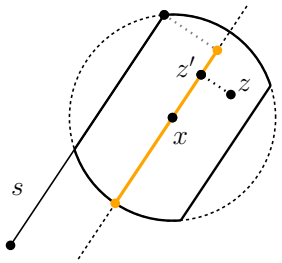


Figure 4 The spine construction used in Lemma 5.

obvious for the shrunken cylindrical shell. For the spherical caps, we note that they are at least as far from x as the cylindrical shell, so the spherical caps of the shrunken capsule are displaced by at least the same amount as the shrunken cylindrical shell. Since only s_1 may be closer to z than $\partial\mathcal{C}(x,r)$, we have $\phi(z) \geq (1-\lambda)r$.

For the upper bound, we consider $\mathcal{C}_{s_1}(x,r)$ in addition to $\mathcal{C}_{s_2}(x,r)$. For any $z \in \mathcal{C}_{s_1}(x,r) \cap \mathcal{C}_{s_2}(x,r)$, let z' and z'' denote the projections of z onto the spines of $\mathcal{C}_{s_1}(x,r)$ and $\mathcal{C}_{s_2}(x,r)$, respectively. By the above derivations $\text{dist}(z,s_2) \leq \text{dist}(z,z'') + \text{dist}(z'',s_2) \leq (1+\lambda)r$. We also have $\text{dist}(z',s_1) = \text{dist}(x,s_1) \leq \phi(x) = r$, implying $\text{dist}(z,s_1) \leq \text{dist}(z,z') + \text{dist}(z',s_1) \leq (1+\lambda)r$. It follows that,

$$\phi(z) \;\leq\; \max\{\text{dist}(z,s_1), \text{dist}(z,s_2)\} \;\leq\; (1+\lambda)r,$$

as desired. ◀

From the above lemma, it is easy to obtain an approximation of nearest-neighbor distances using the segment closest to the center point of the capsule as a representative. This qualifies capsules to serve as cells for a type of approximate Voronoi diagram (AVD) data structure [13,31].

▶ **Lemma 6.** *For any $z \in \mathcal{C}^\lambda(x,\phi(x))$ with $0 < \lambda < 1$, $\text{dist}(z,s_x) \leq \frac{1+\lambda}{1-\lambda}\,\text{dist}(z,s_z)$, where s_x is the closest segment to x and s_z is the closest segment to z.*

Proof. Assume s_z is distinct from s_x, for the assertion holds trivially otherwise. In the same notation we used to prove Lemma 5, let $r = \phi(x)$, and denote by s_x and s'_x the nearest and second-nearest segments to x, respectively. Observing that $r = \text{dist}(x,s'_x)$, the interior of $\mathcal{C}(x,r)$ cannot intersect any segment except for s_x. As seen in the proof of Lemma 5, we have that $\text{dist}(z,s_z) \geq \text{dist}(z,\partial\mathcal{C}(x,r)) \geq (1-\lambda)r$ while $\text{dist}(z,s_x) \leq (1+\lambda)r$. ◀

As an immediate corollary, we have the following.

▶ **Corollary 7.** *For $0 < \varepsilon \leq 1$ and any $z \in \mathcal{C}^\lambda(x,\phi(x))$ with $0 < \lambda \leq \frac{\varepsilon}{3}$, the nearest neighbor of x is a $(1+\varepsilon)$-approximate nearest neighbor of z.*

Capsules enjoy a number of properties (described later in this section) that make them suitable for forming hierarchical covers of space. However, as the intersection of n cylinders and/or balls, capsules can have high combinatorial complexity. For this reason, we use their John ellipsoids in our data structure instead, as in related data structures based on Macbeath regions [1,10]. For $x \in \mathbb{R}^d$ and positive scalars λ and r, define $E^\lambda(x,r)$ as the maximum volume ellipsoid enclosed within $\mathcal{C}^\lambda(x,r)$. By John's Theorem [35], $E^\lambda(x,r) \subseteq \mathcal{C}^\lambda(x,r) \subseteq E^{\lambda\sqrt{d}}(x,r)$. Hence, up to constant factors, these *capsule ellipsoids* can serve as low-complexity proxies for capsule. Our construction makes use of two particular constant scale factors independent of ε, $0 < \lambda'' < \lambda' < 1$. For any x and r, define $E''(x,r) = E^{\lambda''}(x,r)$ and $E'(x,r) = E^{\lambda'}(x,r)$.

3.2 Net-Like Properties for Capsules

In this section, present a number of properties of capsules, demonstrating that they possess similar properties to nets, which arise in the study of metric spaces [33, 39, 40]. While we prove these results for capsules, they all hold for capsule ellipsoids, subject to an adjustment of constant factors.

Our first result is a utility that relates two methods for growing capsules, first by expanding the distance parameter and second by applying a scale factor.

▶ **Lemma 8.** *For any $\gamma \geq 1$, $\mathcal{C}(x, \gamma r) \subseteq \mathcal{C}^{\gamma}(x, r)$, and for $0 < \lambda \leq 1$, $\mathcal{C}^{\lambda}(x, r) \subseteq \mathcal{C}(x, \lambda r)$.*

Proof. For each segment s, the radii used in $\mathcal{C}_s(x, r)$ are of the form $\max(r, \mathrm{dist}(x, s))$. Clearly, $\max(\gamma r, \mathrm{dist}(x, s)) \leq \gamma \cdot \max(r, \mathrm{dist}(x, s))$. The other inequality is similar. ◀

The next lemma bounds the growth in the volume of capsules upon scaling. The first part follows directly from the fact that capsules are full dimensional and convex, and the second part follows from this in combination with Lemma 8.

▶ **Lemma 9.** *For any set of disjoint line segments $S \subseteq \mathbb{R}^d$ and an arbitrary $x \in \mathbb{R}^d$:*

(i) *For $\lambda > 0$, $\mathrm{vol}(\mathcal{C}^{\lambda}(x, r)) = \lambda^d \cdot \mathrm{vol}(\mathcal{C}(x, r))$.*

(ii) *For $\beta \geq 1$, $\mathrm{vol}(\mathcal{C}(x, \beta r)) \leq \beta^d \cdot \mathrm{vol}(\mathcal{C}(x, r))$.*

Next, we derive a packing bound on the number of pairwise interior-disjoint capsules that may fit within a larger capsule.

▶ **Lemma 10.** *Given a set of disjoint line segments $S \subseteq \mathbb{R}^d$, $r \geq 0$, and two constant scale factors $0 < \lambda_p < \lambda_c < 1$, let $Y \subset \mathbb{R}^d$ denote a set of points such that the associated regions $\mathcal{C}^{\lambda_p}(y, r)$, with $y \in Y$, are disjoint. Then, for any $x \in \mathbb{R}^d$ and $\beta \geq 1$, the number of regions in $R_Y = \{\mathcal{C}^{\lambda_c}(y, r) \mid y \in Y\}$ that intersect $\mathcal{C}^{\lambda_c}(x, \beta r)$ is $O(\beta^d)$.*

Proof. Fix a $y \in Y$ such that $\mathcal{C}^{\lambda_c}(x, \beta r) \cap \mathcal{C}^{\lambda_c}(y, r) \neq \emptyset$. As $\mathcal{C}^{\lambda_c}(y, r) \subseteq \mathcal{C}^{\lambda_c}(y, \beta r)$, we also have that $\mathcal{C}^{\lambda_c}(x, \beta r) \cap \mathcal{C}^{\lambda_c}(y, \beta r) \neq \emptyset$. Applying Lemma 4 (with the roles of x and y swapped), we obtain $\mathcal{C}^{\lambda_c}(x, \beta r) \subseteq \mathcal{C}^{\alpha\lambda_c}(y, \beta r)$, where $\alpha = \frac{3+\lambda_c}{1-\lambda_c} > 1$. Lemma 9 yields

$$\begin{aligned}\mathrm{vol}(\mathcal{C}^{\lambda_p}(y, r)) &= \left(\frac{\lambda_p}{\lambda_c}\right)^d \mathrm{vol}(\mathcal{C}^{\lambda_c}(y, r)) \geq \left(\frac{\lambda_p}{\lambda_c \beta}\right)^d \mathrm{vol}(\mathcal{C}^{\lambda_c}(y, \beta r)) \\ &= \left(\frac{\lambda_p}{\lambda_c \alpha \beta}\right)^d \mathrm{vol}(\mathcal{C}^{\alpha\lambda_c}(y, \beta r)) \geq \left(\frac{\lambda_p}{\lambda_c \alpha \beta}\right)^d \mathrm{vol}(\mathcal{C}^{\lambda_c}(x, \beta r)).\end{aligned}$$

By packing, the number of regions of R_Y intersecting $\mathcal{C}^{\lambda_c}(x, \beta r)$ is $O\big(\big(\frac{\lambda_c \alpha \beta}{\lambda_p}\big)^d\big)$. The result follows since λ_c, λ_p, and α are all constants. ◀

Turning our attention to radius assignment through the local feature size ϕ, we show that expansion-containment still holds.

▶ **Lemma 11.** *Given two points $x, y \in \mathbb{R}^d$ and $0 < \lambda < 1$, if $\mathcal{C}^{\lambda}(x, \phi(x)) \cap \mathcal{C}^{\lambda}(y, \phi(y)) \neq \emptyset$, then $\mathcal{C}^{\lambda}(y, \phi(y)) \subseteq \mathcal{C}^{\beta\lambda}(x, \phi(x))$ for a constant $\beta = \frac{(3+\lambda)(1+\lambda)}{(1-\lambda)^2}$.*

Proof. Assuming first that $\phi(y) \leq \phi(x)$, $\mathcal{C}^{\lambda}(y, \phi(y)) \subseteq \mathcal{C}^{\lambda}(y, \phi(x))$ and thus

$$\mathcal{C}^{\lambda}(x, \phi(x)) \cap \mathcal{C}^{\lambda}(y, \phi(y)) \neq \emptyset \implies \mathcal{C}^{\lambda}(x, \phi(x)) \cap \mathcal{C}^{\lambda}(y, \phi(x)) \neq \emptyset.$$

Applying Lemma 4 with $r = \phi(x)$, implies that $\mathcal{C}^{\lambda}(y, \phi(y)) \subseteq \mathcal{C}^{\alpha\lambda}(x, \phi(x))$.

Otherwise, $\phi(x) \leq \phi(y)$, and by applying Lemma 5 twice with any $z \in \mathcal{C}^{\lambda}(x, \phi(x)) \cap \mathcal{C}^{\lambda}(y, \phi(y))$, we obtain

$$\phi(y) \;\leq\; \frac{1}{1-\lambda}\phi(z) \;\leq\; \frac{1+\lambda}{1-\lambda}\phi(x).$$

By Lemma 8, $\mathcal{C}^{\lambda}(x, \gamma\,\phi(x)) \subseteq \mathcal{C}^{\gamma\lambda}(x, \phi(x))$, where $\gamma = \frac{1+\lambda}{1-\lambda} > 1$. Therefore

$$\mathcal{C}^{\lambda}(x, \phi(x)) \cap \mathcal{C}^{\lambda}(y, \phi(y)) \neq \emptyset \implies \mathcal{C}^{\lambda}(x, \gamma \cdot \phi(x)) \cap \mathcal{C}^{\lambda}(y, \gamma \cdot \phi(x)) \neq \emptyset.$$

Applying Lemma 4 with $r = \gamma\,\phi(x)$, implies that $\mathcal{C}^{\lambda}(y, \phi(y)) \subseteq \mathcal{C}^{\alpha\gamma\lambda}(x, \phi(x))$. ◀

4 The ANN Data Structure

In this section, we apply the results of the previous section to present our data structure for answering ε-ANN queries. Again, $S = \{s_1, \ldots, s_n\}$ is a set of n disjoint line segments in $\mathbb{R}^d$, and $\varepsilon > 0$ is the approximation parameter. Let $B(S)$ be a minimum volume Euclidean ball that contains S, and let $B^+(S)$ denote a concentric expansion of $B(S)$ about its center by a factor of $1 + 2/\varepsilon$. It is easy to see that if the query point q lies outside of $B^+(S)$, any segment may be reported as an ε-ANN of q. Thus, for the rest of the construction, we focus on query points lying within $B^+(S)$. Let x_0 and r^+ denote the center and radius of this ball, respectively. Clearly, $r^+ = \Theta(\mathrm{diam}(S)/\varepsilon)$. Let $\delta(S)$ denote the minimum distance between any two segments of S. Observe that for every point $x \in B^+(S)$, its local feature size, $\phi(x)$, is at least $\delta(S)/2$.

Here is a high-level overview of the data structure. It consists of a rooted directed acyclic graph (DAG), which is based on covering $B^+(S)$ with a hierarchy of capsule ellipsoids of exponentially diminishing scales. The DAG is organized in levels, with a single root node at level zero whose associated capsule ellipsoid contains $B^+(S)$. For $i \geq 0$, the capsule ellipsoids associated with the nodes of level i employ the scale parameter $r_i = r^+/2^i$, and thus successive levels are more refined. Each level of the DAG will be associated with a collection of capsule ellipsoids that cover $B^+(S)$. In particular, each node at level i of the DAG stores a point $x \in B^+(S)$, and the associated capsule ellipsoid, denoted $E'(x)$, centered at this point is defined to be the shrunken ellipsoid $E'(x, r_i)$ with respect to S. (The shrinking factor $\frac{1}{2}$ can be taken to be any constant smaller than 1, subject to an adjustment in the various constant factors used in our construction.) Each node at level i will either be declared to be a leaf, or it will be linked to those nodes at level $i+1$ whose capsule ellipsoids it overlaps, which we call its *children*. (We will show that the out-degree of any node is a constant.)

We continue this refinement process until $r_i \leq \frac{\varepsilon}{3}\phi(x)$. The resulting terminal nodes are called *leaves*, and each stores the segment of S that is closest to the capsule center as its *representative*.

Queries are answered by a simple descent through this DAG. Assuming that the query point q lies within $B^+(S)$, we descend level by level through the DAG. On arriving at a non-leaf node, we inspect the ellipsoids of its children on level $i+1$. Their associated capsules cover $E'(x)$, and the search continues with any one of these children whose associated ellipsoid contains q. When the search arrives at a leaf node, the associated representative segment is returned as the answer to the query.

The DAG is constructed in a top-down manner, starting with the root of the DAG. The root capsule is $E'(x_0)$, where x_0 is the center of $B^+(S)$. We assert that this covers $B^+(S)$. To see this, observe that by definition, $E'(x, r)$ contains the ball of radius r centered at x, and hence the same holds for $E'(x, r)$. For $i = 1, 2, \ldots$, let U_i denote the portion of $B^+(S)$ that

is not covered by any of the leaves of the structure from prior levels. For any $x \in \mathbb{R}^d$, define $E''(x) = E''(x, r_i)$. Let X_i be any maximal set of points x within U_i such that the associated ellipsoids $E''(x)$ are pairwise disjoint. It follows from maximality and expansion-containment that the union of the expanded ellipsoids $E'(x)$ for $x \in X_i$ covers U_i. We create a node at level i of the DAG for each point $x \in X_i$, and we link each such node as a child of any non-leaf node from the previous level whose E' capsule ellipsoid (computed with respect to level $i-1$) it overlaps.

In the remainder of this section, we analyze the correctness, query time and storage requirements of this data structure. Our first two lemmas establish correctness and bound the depth of the data structure. Correctness follows from Corollary 7.

▶ **Lemma 12.** *Given a set S of line segments, the above search algorithm returns an ε-ANN among the segments of S for any query point $q \in B^+(S)$.*

Next we analyze the depth of the DAG.

▶ **Lemma 13.** *Given a set S of n disjoint segments in $\mathbb{R}^d$ with spread Δ, the ε-AVD structure described above has $O(\log(\max(n, \Delta)/\varepsilon))$ levels.*

Proof. Recall that $B(S)$ is the minimum Euclidean ball containing S and $B^+(S)$ is its expansion by $1 + \frac{2}{\varepsilon}$, and r^+ is its radius. Clearly, $r^+ = \Theta(\mathrm{diam}(S)/\varepsilon)$. Letting $\delta_{\min}$ denote the minimum distance between any pair of segments, $\Delta = \mathrm{diam}(S)/\delta_{\min}$. Clearly, for any point x, $\phi(x) \geq \delta_{\min}$. The refinement process terminates when the scale falls below $\frac{\varepsilon}{3}\phi(x) \geq \frac{\varepsilon}{3}\delta_{\min}$. Since the scale decreases by a factor of 2 with each level of the data structure, the total number of levels is

$$O\left(\log \frac{r^+}{\varepsilon\,\delta_{\min}}\right) = O\left(\log \frac{\mathrm{diam}(S)}{\varepsilon^2 \delta_{\min}}\right) = O\left(\log \frac{\Delta}{\varepsilon}\right),$$

as desired. Recall that the spread of a set of segments in $\mathbb{R}^d$ grows at least polynomially with n, therefore $\log n$ is $O(\log \Delta)$. ◀

Our next result bounds the number of children for each node.

▶ **Lemma 14.** *Each non-leaf node of the data structure has $O(1)$ children.*

Proof. Consider a node at some level i centered at a point x. Let $r = r_{i+1} = r_i/2$. This node's children consist of the nodes y of level $i+1$ whose ellipsoid $E'(y, r_{i+1})$ overlaps $E'(x, r_i)$. By construction, all such points y come from a set Y whose ellipsoids $E''(y, r_{i+1})$ are disjoint. By applying Lemma 10 in the elliptical setting, the number of such overlapping ellipsoids is $O(2^d) = O(1)$, given our assumption that the dimension d is fixed. ◀

Since each node of the DAG has constant degree, it follows that the overall query time is proportional to DAG's height, which is $O(\log(\Delta/\varepsilon))$. Finally, we bound the total space used by the data structure.

▶ **Lemma 15.** *Given a set S of n line segments in $\mathbb{R}^d$, the total storage required by the ε-AVD is $O((n^2/\varepsilon^d)\log\frac{\Delta}{\varepsilon})$.*

Proof. We distinguish between two transitions within the DAG structure. When the scale parameter r_i of a node x first falls below $\phi(x)$, we say that this is a basic leaf, and when it falls below $\frac{\varepsilon}{3}\phi(x)$ (the actual termination condition), we say it is a final leaf. Lemma 17 (presented in Section 5) states that the number of capsules at the basic leaf level that are charged to any pair of segments is $O(\log\frac{\Delta}{\varepsilon})$. Therefore, the total number basic leaves is $O(n^2 \log\frac{\Delta}{\varepsilon})$.

Observe that all of the points lying within $E'(x, \phi(x))$ share the same local-feature size values up to constant factors, and therefore, all the descendants of this node lie within the next $O(\log \frac{1}{\varepsilon})$ levels of the structure. They are all similarly shaped (up to constant factors), but their sizes are smaller by a factor of at least $\frac{\varepsilon}{3}$. By the disjointness of the shrunken E'' capsule ellipsoids, it follows that the number of descendants is $O(1/\varepsilon^d)$. Therefore, the total number of nodes in the DAG is $O((n^2/\varepsilon^d) \log \frac{\Delta}{\varepsilon})$. ◀

By combining the results of the previous lemmas, we obtain Theorem 1.

5 Storage Bounds for Capsules

Recall the definition of capsules from Section 3. The infinite cylindrical component in the definition of $\mathcal{C}(x, r)$ induced by a particular segment $s_i \in S$ will be denoted $\mathrm{Cyl}_i(x, r)$ and its radius will be denoted by $t_i(x)$.

For a given point $x \in \mathbb{R}^d$ and $r > 0$, we distinguish two such cylinders. Without loss of generality, let s_1 denote the nearest neighbor of x in S. Letting v_1 be a unit vector parallel to s_1 (the direction does not matter by central symmetry). Denote by ℓ_x the line passing through x in the direction of v_1. Using ℓ_x, we define the set of points p_i as the intersection of $\partial \mathrm{Cyl}_i(x, r)$ and ℓ_x such that $\langle v_1, p_i - x \rangle > 0$. Again, without loss of generality, let p_2 denote the closest intersection point to x, such that $\mathrm{Cyl}_2(x, r)$ is the cylinder generating p_2. We use the two cylinders $\mathrm{Cyl}_1(x, r)$ and $\mathrm{Cyl}_2(x, r)$ to sandwich the capsule $\mathcal{C}(x, r)$ between two simple shapes providing lower and upper bounds on its volume. The radii of $\mathrm{Cyl}_1(x, r)$ and $\mathrm{Cyl}_2(x, r)$ will be denoted by t_1 and t_2, respectively. By definition, $t_1 = r$. When r is chosen as the local feature size $\phi(x)$ at x, we also have $t_2 \geq r$.

The inner bounding volume $V^-(x, r)$ is defined as the double cone whose axis is ℓ_x and base is the $(d-1)$-dimensional disk of radius r centered at x orthogonal to ℓ_x, with two apexes at p_2 and (symmetrically about x) $2x - p_2$. The outer bounding volume $V^+(x, r)$ is defined as the cylinder with ℓ_x as axis whose radius is r and height is equal to the length of the projection of $\mathrm{Cyl}_1(x, r) \cap \mathrm{Cyl}_2(x, r)$ onto ℓ_x.

▶ **Lemma 16.** *Fix a point $x \in \mathbb{R}^d$ and let $C(x, r)$ be the capsule of radius r induced at x by a set S of n line segments. Then,*

$$V^-(x, r) \subseteq \mathcal{C}(x, r) \subseteq V^+(x, r), \quad \text{and} \quad \frac{\mathrm{vol}(V^+(x, r))}{\mathrm{vol}(V^-(x, r))} \leq 2(d+1).$$

Proof. The containment follows by the construction of the bounding volumes. In bounding the ratio of the two volumes, we use the same notation and assumptions as above, without loss of generality. Letting θ denote the acute angle between the two lines supporting s_1 and s_2, and $\mathbb{V}_{d-1}$ denote the volume of a unit ball in $\mathbb{R}^{d-1}$, we have

$$\frac{\mathrm{vol}(V^+(x, r))}{\mathrm{vol}(V^-(x, r))} \leq \frac{4\mathbb{V}_{d-1} r^{d-1} t_1(x) \cdot \csc(\theta)}{\frac{2}{d+1}\mathbb{V}_{d-1} r^{d-1} t_1(x) \cdot \csc(\theta)} \leq 2 \cdot (d+1),$$

where the numerator is the volume of the intersection of two cylinders, and the denominator is the volume of a cone in $\mathbb{R}^d$. ◀

In order to bound the number of leaf-level capsules within a ball of radius $O\left(\frac{1}{\varepsilon} \cdot \mathrm{diam}(S)\right)$, we use the following charging scheme. Again, we use the simplified notation and assumptions from before. A capsule $\mathcal{C}(x, r)$ will be charged to the two line segments s_1 and s_2. For a fixed pair of segments s_i and s_j, acting as respectively as s_1 and s_2 for the point x in consideration, we may restrict attention to all center points x lying in the cylinder of radius r with the line supporting s_i as axis; denote this cylinder by $\mathrm{Cyl}(s_i, r)$.

▶ **Lemma 17.** *The number of leaf level capsules charged to any pair of segments is* $O(\log \frac{\Delta}{\varepsilon})$.

Proof. We cover the points in $\mathrm{Cyl}(s_i, r)$ using a sequence of growing cylindrical intervals based on their distances to s_j. Define $\mathcal{I}_k(s_i, r)$ to be the set of points in $\mathrm{Cyl}(s_i, r)$ such that for all $x \in \mathcal{I}_k(s_i, r)$ we have that $2^{k-1}r \leq \mathrm{dist}(x, s_j) \leq 2^k r$. In other words, $\mathcal{I}_k(s_i, r)$ is the intersection of $\mathrm{Cyl}(s_i, r)$ with the *cylindrical shell* $\mathrm{Cyl}(s_j, 2^k r) \setminus \mathrm{Cyl}(s_j, 2^{k-1}r)$. It is easy to see that the volume of the intersection is maximized when s_i intersects s_j due to the symmetry of the cylinder $\mathrm{Cyl}(s_i, r)$ about the line supporting s_i. Similar to the upper bound on the volume of $\mathcal{C}(x, r)$ by that of $V^+(x, r)$, we see that $\mathrm{vol}(\mathcal{I}_k(s_i, r))$ is at most $4\mathbb{V}_{d-1}2^k r^d \cdot \csc(\theta)$, where θ is the acute angle between the two lines supporting s_i and s_j. For any capsule charged to s_i and s_j with center $x \in \mathcal{I}_k(s_i, r)$, we use the inner volume $V^-(x, r) \subseteq \mathcal{C}(x, r)$ to obtain a lower bound $\mathrm{vol}(C^\lambda(x, r)) \geq \frac{1}{d+1}\mathbb{V}_{d-1}2^k r^d \lambda^d \cdot \csc(\theta)$. By choosing the capsule centers to have global packing-covering properties as a Delone set, it follows that there can be at most $4(d+1)/\lambda^d = O(1)$ capsules centered within $\mathcal{I}_k(s_i, r)$.

The desired bound follows by repeating the above argument over all scales r, a total of $\log(\Delta)$, and all k with $2^k r \leq \frac{1}{\varepsilon} \cdot \mathrm{diam}(S)$, a total of $\log(1/\varepsilon)$. ◀

6 Conclusions and Future Work

We have presented a new AVD-based approach to answering ε-segment ANN queries based on a hierarchical covering of space by ellipsoids. By elaborating on the intrinsic geometry underlying more general distance functions, our work helps pave the way to extend well-established techniques from data structure design and approximation algorithms in Euclidean and metric spaces to more general geometries. Specifically, we anticipate further progress in understanding the metric-like structure defined by the local tensors derived from the Hessians of distance functions. This should directly benefit the development of more efficient space covers, e.g., circumventing dependence on the spread and other geometric parameters. In addition, we expect polytope approximation techniques to enable better ε-dependencies in the storage requirements. Put together, we leave it to future work to achieve both remaining tasks of eliminating dependence on the spread and improving low-level query processing to obtain $O(\log n/\varepsilon)$ query times with only $O(n^2/\varepsilon^{d/2})$ storage, by analogy with the best known corresponding results for ANN against point sets under the Euclidean metric modulo the quadratic dependence on n needed for storage in the case of segment-ANN.

References

1. A. Abdelkader and D. M. Mount. Economical Delone sets for approximating convex bodies. In *Proc. 16th Scand. Workshop Algorithm Theory*, pages 4:1–4:12, 2018.
2. P. K. Agarwal, N. Rubin, and M. Sharir. Approximate nearest neighbor search amid higher-dimensional flats. In *Proc. 25th Annu. European Sympos. Algorithms*, volume 87, pages 4:1–4:13, 2017.
3. O. Aichholzer, D. Z. Chen, D. T. Lee, A. Mukhopadhyay, E. Papadopoulou, and F. Aurenhammer. Voronoi diagrams for direction-sensitive distances. In *Proc. 13th Annu. Sympos. Comput. Geom.*, pages 418–420, 1997.
4. A. Andoni, P. Indyk, R. Krauthgamer, and H. L. Nguyên. Approximate line nearest neighbor in high dimensions. In *Proc. 20th Annu. ACM-SIAM Sympos. Discrete Algorithms*, pages 293–301, 2009.
5. A. Andoni, A. Naor, A. Nikolov, I. Razenshteyn, and E. Waingarten. Hölder homeomorphisms and approximate nearest neighbors. In *Proc. 59th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 159–169, 2018.

6 Boris Aronov. A lower bound on Voronoi diagram complexity. *Inform. Process. Lett.*, 83(4):183–185, 2002.

7 S. Arya, G. D. da Fonseca, and D. M. Mount. Near-optimal ε-kernel construction and related problems. In *Proc. 33rd Internat. Sympos. Comput. Geom.*, pages 10:1–15, 2017. arXiv:1604.01175.

8 S. Arya, G. D. da Fonseca, and D. M. Mount. On the combinatorial complexity of approximating polytopes. *Discrete Comput. Geom.*, 58(4):849–870, 2017. doi:10.1007/s00454-016-9856-5.

9 S. Arya, G. D. da Fonseca, and D. M. Mount. Optimal approximate polytope membership. In *Proc. 28th Annu. ACM-SIAM Sympos. Discrete Algorithms*, pages 270–288, 2017.

10 S. Arya, G. D. da Fonseca, and D. M. Mount. Approximate convex intersection detection with applications to width and Minkowski sums. In *Proc. 26th Annu. European Sympos. Algorithms*, pages 3:1–14, 2018. doi:10.4230/LIPIcs.ESA.2018.3.

11 S. Arya, G. D. da Fonseca, and D. M. Mount. Approximate polytope membership queries. *SIAM J. Comput.*, 47(1):1–51, 2018. doi:10.1137/16M1061096.

12 S. Arya and T. Malamatos. Linear-size approximate Voronoi diagrams. In *Proc. 13th Annu. ACM-SIAM Sympos. Discrete Algorithms*, pages 147–155, 2002.

13 S. Arya, T. Malamatos, and D. M. Mount. Space-time tradeoffs for approximate nearest neighbor searching. *J. Assoc. Comput. Mach.*, 57:1–54, 2009. doi:10.1145/1613676.1613677.

14 F. Aurenhammer, B. Jüttler, and G. Paulini. Voronoi Diagrams for Parallel Halflines and Line Segments in Space. In *Proc. 28th Annu. Internat. Sympos. Algorithms Comput.*, volume 92, pages 7:1–7:10, 2017.

15 I. Bárány. The technique of M-regions and cap-coverings: A survey. *Rend. Circ. Mat. Palermo*, 65:21–38, 2000.

16 G. Barequet, E. Papadopoulou, and M. Suderland. Unbounded Regions of High-Order Voronoi Diagrams of Lines and Segments in Higher Dimensions. In *Proc. 30th Annu. Internat. Sympos. Algorithms Comput.*, volume 149, pages 62:1–62:15, 2019.

17 R. Basri, T. Hassner, and L. Zelnik-Manor. Approximate nearest subspace search. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33:266–278, 2011.

18 J.-D. Boissonnat, M. Sharir, B. Tagansky, and M. Yvinec. Voronoi diagrams in higher dimensions under certain polyhedral distance functions. *Discrete Comput. Geom.*, 19(4):485–519, April 1998.

19 B. Chazelle, D. Liu, and A. Magen. Approximate range searching in higher dimension. *Comput. Geom. Theory Appl.*, 39:24–29, 2008.

20 L.P. Chew, K. Kedem, M. Sharir, B. Tagansky, and E. Welzl. Voronoi diagrams of lines in 3-space under polyhedral convex distance functions. *J. Algorithms*, 29(2):238–255, 1998.

21 S. Cost and S. Salzberg. A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning*, 10:57–78, 1993.

22 T. M. Cover and P. E. Hart. Nearest neighbor pattern classification. *IEEE Trans. Image Proc.*, 13:57–67, 1967.

23 S. Deerwester, S. T. Dumals, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *J. Amer. Soc. Inform. Sci.*, 41:391–407, 1990.

24 L. Devroye and T. J. Wagner. Nearest neighbor methods in discrimination. In P. R. Krishnaiah and L. N. Kanal, editors, *Handbook of Statistics*, volume 2. North-Holland, 1982.

25 R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* John Wiley & Sons, NY, 1973.

26 I. Z. Emiris, T. Malamatos, and E. Tsigaridas. Approximate nearest neighbor queries among parallel segments. In *26th European Workshop on Computational Geometry (EuroCG 2010)*, 2010.

27 H. Everett, D. Lazard, S. Lazard, and M. Safey El Din. The Voronoi diagram of three lines. *Discrete Comput. Geom.*, 42(1):94–130, July 2009.

28 U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy. *Advances in Knowledge Discovery and Data Mining.* AAAI Press/Mit Press, 1996.

29 M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by image and video content: The QBIC system. *IEEE Computer*, 28:23–32, 1995.

30 A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic, Boston, MA, 1991.

31 S. Har-Peled. A replacement for Voronoi diagrams of near linear size. In *Proc. 42nd Annu. IEEE Sympos. Found. Comput. Sci.*, pages 94–103, 2001.

32 S. Har-Peled, P. Indyk, and R. Motwani. Approximate nearest neighbor: Towards removing the curse of dimensionality. *Theo. of Comput.*, 8:321–350, 2012.

33 S. Har-Peled and M. Mendel. Fast construction of nets in low dimensional metrics, and their applications. *SIAM J. Comput.*, 35:1148–1184, 2006.

34 P. Indyk. Nearest neighbors in high-dimensional spaces. In J. E. Goodman and J. O'Rourke, editors, *The Handbook of Discrete and Computational Geometry, 2nd Edition*, pages 877–892. Chapman & Hall/CRC, Boca Raton, FL, 2004.

35 F. John. Extremum problems with inequalities as subsidiary conditions. In *Studies and Essays Presented to R. Courant on his 60th Birthday*, pages 187–204. Interscience Publishers, Inc., New York, 1948.

36 M. I. Karavelas. A robust and efficient implementation for the segment Voronoi diagram. In *Internat. Symp. on Voronoi Diagrams in Sci. and Engin.*, volume 2004, pages 51–62, 2004.

37 V. Koltun and M. Sharir. 3-dimensional Euclidean Voronoi diagrams of lines with a fixed number of orientations. *SIAM J. Comput.*, 32(3):616–642, 2003.

38 V. Koltun and M. Sharir. Polyhedral Voronoi diagrams of polyhedra in three dimensions. *Discrete Comput. Geom.*, 31(1):83–124, January 2004.

39 R. Krauthgamer and J. R. Lee. Navigating nets: Simple algorithms for proximity search. In *Proc. 15th Annu. ACM-SIAM Sympos. Discrete Algorithms*, pages 798–807, 2004.

40 R. Krauthgamer and J. R. Lee. The black-box complexity of nearest-neighbor search. *Theo. Comp. Sci.*, 348:262–276, 2005.

41 A. Laddha, Y. T. Lee, and S. Vempala. Strong self-concordance and sampling. In *Proc. 52nd Annu. ACM Sympos. Theory Comput.*, pages 1212–1222, 2020.

42 A. M. Macbeath. A compactness theorem for affine equivalence-classes of convex regions. *Canad. J. Math*, 3:54–61, 1950.

43 S. Mahabadi. Approximate nearest line search in high dimensions. In *Proc. 26th Annu. ACM-SIAM Sympos. Discrete Algorithms*, pages 337–354, 2015.

44 W. Mulzer, H. L. Nguyên, P. Seiferth, and Y. Stein. Approximate k-flat nearest neighbor search. In *Proc. 47th Annu. ACM Sympos. Theory Comput.*, pages 783–792, 2015.

45 H. Narayanan. Randomized interior point methods for sampling and optimization. *Ann. Appl. Probab.*, 26(1):597–641, 2016.

46 Y. Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Springer Publishing Company, Incorporated, 2014.

47 Y. Nesterov and M. Todd. On the Riemannian geometry defined by self-concordant barriers and interior-point methods. *Found. Comput. Math.*, 2(4):333–361, 2002.

48 S. Sachdeva and N. K. Vishnoi. The mixing time of the Dikin walk in a polytope—A simple proof. *Oper. Res. Lett.*, 44(5):630–634, 2016.

49 K. Sugihara and M. Iri. A robust topology-oriented incremental algorithm for Voronoi diagrams. *Internat. J. Comput. Geom. Appl.*, 04:179–228, 1994.

50 S. Vijayanarasimhan, P. Jain, and K. Grauman. Hashing hyperplane queries to near points with applications to large-scale active learning. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36:276–288, 2014.

51 X. Wang, S. Atev, J. Wright, and G. Lerman. Fast subspace search via Grassmannian based hashing. In *Proc. IEEE Internat. Conf. Comput. Vision (ICCV)*, December 2013.

52 S. J. Wright. *Primal-Dual Interior-Point Methods*. Society for Industrial and Applied Mathematics, 1997.

Chasing Puppies: Mobile Beacon Routing on Closed Curves

Mikkel Abrahamsen
BARC, University of Copenhagen, Denmark

Jeff Erickson
University of Illinois at Urbana-Champaign, IL, USA

Irina Kostitsyna
Eindhoven University of Technology, The Netherlands

Maarten Löffler
Utrecht University, The Netherlands

Tillmann Miltzow
Utrecht University, The Netherlands

Jérôme Urhausen
Utrecht University, The Netherlands

Jordi Vermeulen
Utrecht University, The Netherlands

Giovanni Viglietta
Japan Advanced Institute of Science and Technology, Nomi City, Ishikawa, Japan

Abstract

We solve an open problem posed by Michael Biro at CCCG 2013 that was inspired by his and others' work on beacon-based routing. Consider a human and a puppy on a simple closed curve in the plane. The human can walk along the curve at bounded speed and change direction as desired. The puppy runs with unbounded speed along the curve as long as the Euclidean straight-line distance to the human is decreasing, so that it is always at a point on the curve where the distance is locally minimal. Assuming that the curve is smooth (with some mild genericity constraints) or a simple polygon, we prove that the human can always catch the puppy in finite time.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Beacon routing, navigation, generic smooth curves, puppies

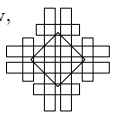
Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.5

Related Version *Full Version*: https://arxiv.org/abs/2103.09811

Supplementary Material *Software (Source Code)*:
https://github.com/viglietta/Chasing-Puppies
archived at swh:1:dir:58dd270b0896aa11024666b5cbd2481068e8eab9

Funding *Mikkel Abrahamsen*: Partially supported by the VILLUM Foundation grant 16582.
Maarten Löffler: Partially supported by the Dutch Research Council (NWO) under project number 614.001.504 and 628.011.005.
Tillmann Miltzow: Supported by the Dutch Research Council (NWO) under Veni grant EAGER.
Jérôme Urhausen: Supported by the Dutch Research Council (NWO); 612.001.651.
Jordi Vermeulen: Supported by the Dutch Research Council (NWO); 612.001.651.

Acknowledgements The authors wish to thank the anonymous reviewers for useful comments and suggestions. Thanks to Ivor van der Hoog, Marc van Kreveld, and Frank Staals for helpful discussions in the early stages of this work, and to Joseph O'Rourke for clarifying the history of the problem. Portions of this work were done while the second author was visiting Utrecht University.



1 Introduction

You have lost your puppy somewhere on a simple closed curve. Both of you are forced to stay on the curve. You can see each other and both want to reunite. The problem is that the puppy runs infinitely faster than you, and it believes naively that it is always a good idea to minimize its straight-line distance to you. What do you do?

To be more precise, let $\gamma\colon S^1 \hookrightarrow \mathbb{R}^2$ be a simple closed curve in the plane, which we informally call the *track*. Two special points move around the track, called the *puppy* p and the *human* h. The human can walk along the track at bounded speed and change direction as desired. The puppy runs with unbounded speed along the track as long as its Euclidean straight-line distance to the human is decreasing, until it reaches a point on the curve where the distance is locally minimized. As the human moves along the track, the puppy moves to stay at a local distance minimum. The human's goal is to move in such a way that the puppy and the human meet. See Figure 1 for a simple example.

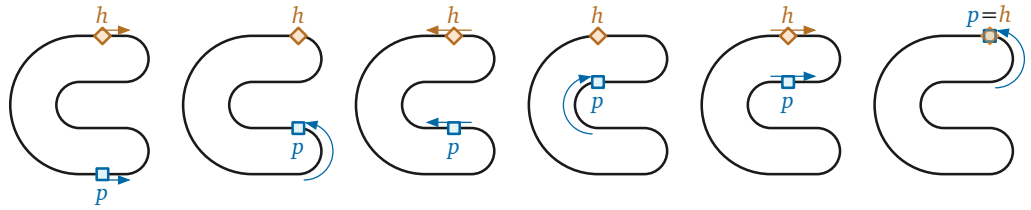


Figure 1 Catching the puppy.

In this paper we show that it is always possible for the human to reunite with the puppy, under the assumption that the curve is well-behaved in a sense to be defined.

This problem was posed in a different guise at the open problem session of the 25th Canadian Conference on Computational Geometry (CCCG 2013) by Michael Biro. In Biro's formulation, the track was a railway, the human a locomotive, and the puppy a train carriage that was attracted to an infinitely strong magnet installed in the locomotive.

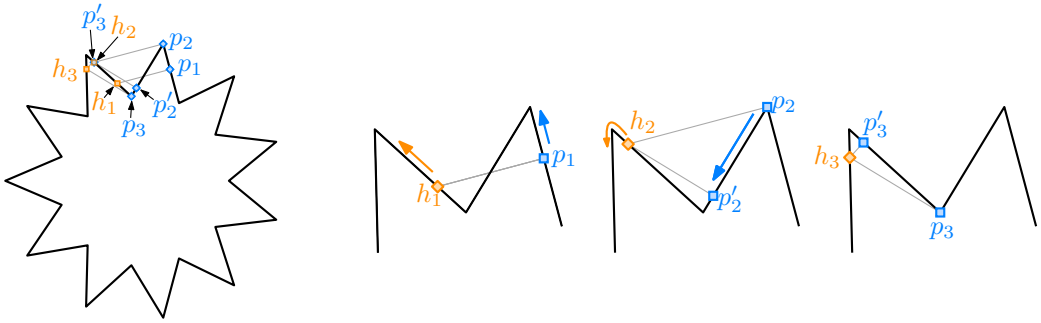


Figure 2 If the human walks only counterclockwise from h_1, the human and the puppy will never meet. To the right are closeups of two of the spikes of the star.

Returning to our formulation of catching a puppy, it was also asked if the human will always catch the puppy by choosing an arbitrary direction and walking only in that direction. This turns out not to be the case; consider the star-shaped track in Figure 2. Suppose the human and puppy start at points h_1 and p_1, respectively, and the human walks counterclockwise around the track. When the human reaches h_2, the puppy runs from p_2 to p'_2. When the human reaches h_3, the puppy runs from p_3 to p'_3. Then the pattern repeats

indefinitely. Examples of this type, where the human walking in the wrong direction will never catch the puppy, were independently discovered during the conference by some of the authors and by David Eppstein.

1.1 Related work

Biro's problem was inspired by his and others' work on *beacon-based geometric routing*, a generalization of both greedy geometric routing and the art gallery problem introduced at the 2011 Fall Workshop on Computational Geometry [8] and the 2012 Young Researchers Forum [9], and further developed in Biro's PhD thesis [7] and papers [10, 11]. A *beacon* is a stationary point object that can be activated to create a "magnetic pull" towards itself everywhere in a given polygonal domain P. When a beacon at point b is *activated*, a point object p moves moves greedily to decrease its Euclidean distance to b, alternately moving through the interior of P and sliding along its boundary, until it either reaches b or gets stuck at a "dead point" where Euclidean distance is minimized. By activating different beacons one at a time, one can route a moving point object through the domain. Initial results for this model by Biro and his colleagues [7–11] sparked significant interest and subsequent work in the community [3, 4, 6, 14, 19, 21–23, 27]. More recent works have also studied how to utilize objects that repel points instead of attracting them [12, 25].

Biro's problem can also be viewed as a novel variant of classical *pursuit* problems, which have been an object of intense study for centuries [26]. The oldest pursuit problems ask for a description of the *pursuit curve* traced by a *pursuer* moving at constant speed directly toward a *target* moving along some other curve. Pursuit curves were first systematically studied by Bouguer [13] and de Maupertuis [15] in 1732, who used the metaphor of a pirate overtaking a merchant ship; another notable example is Hathaway's problem [17], which asks for the pursuit curve of a dog swimming at unit speed in a circular lake directly toward a duck swimming at unit speed around its circumference. In more modern *pursuit-evasion* problems, starting with Rado's famous "lion and man" problem [24, pp.114–117], the pursuer and target both move strategically within some geometric domain; the pursuer attempts to *capture* the target by making their positions coincide while the target attempts to evade capture. Countless variants of pursuit-evasion problems have been studied, with multiple pursuers and/or targets, different classes of domains, various constraints on motion or visibility, different capture conditions, and so on. Biro's problem can be naturally described as a *cooperative pursuit* or *pursuit-attraction* problem, in which a strategic target (the human) *wants* to be captured by a greedy pursuer (the puppy).

Kouhestani and Rappaport [20] studied a natural variant of Biro's problem, which we can recast as follows. A *guppy* is restricted to a closed and simply-connected *lake*, while the human is restricted to the boundary of the lake. The guppy swims with unbounded speed to decrease its Euclidean distance to the human as quickly as possible. Kouhestani and Rappaport described a polynomial-time algorithm that finds a strategy for the human to catch the guppy, if such a strategy exists, given a simple polygon as input; they also conjectured that a capturing strategy always exists. Abel, Akitaya, Demaine, Demaine, Hesterberg, Korman, Ku, and Lynch [1] recently proved that for some polygons and starting configurations, the human cannot catch the guppy, even if the human is allowed to walk in the exterior of the polygon, thereby disproving Kouhestani and Rappaport's conjecture. Their simplest counterexample is an orthogonal polygon with about 50 vertices.

1.2 Our results

Before describing our results in detail, we need to carefully define the terms of the problem. The *track* is a simple closed curve $\gamma\colon S^1 \hookrightarrow \mathbb{R}^2$. We consider the motion of two points on this curve, called the *human* (or *beacon* or *target*) and the *puppy* (or *pursuer*). A *configuration* is a pair $(x, y) \in S^1 \times S^1$ that specifies the locations $h = \gamma(x)$ and $p = \gamma(y)$ for the human and puppy, respectively. Let $D(x, y)$ denote the straight-line Euclidean distance between these two points. When the human is located at $h = \gamma(x)$, the puppy moves from $p = \gamma(y)$ to greedily decrease its distance to the human, as follows.

- If $D(x, y + \varepsilon) < D(x, y)$ for all sufficiently small $\varepsilon > 0$, the puppy runs forward along the track, by increasing the parameter y.
- If $D(x, y - \varepsilon) < D(x, y)$ for all sufficiently small $\varepsilon > 0$, the puppy runs backward along the track, by decreasing the parameter y.

If both of these conditions hold, the puppy runs in an arbitrary direction. While the puppy is running, the human remains stationary. If neither condition holds, the configuration is *stable*; the puppy does not move until the human does. When the configuration is stable, the human can walk in either direction along the track; the puppy walks along the track in response to keep the configuration stable, until it is forced to run again. The human's goal is to *catch* the puppy; that is, to reach a configuration in which the two points coincide.

Our main result is that the human can always catch the puppy in finite time, starting from any initial configuration, provided the track is either a generic simple smooth curve or an arbitrary simple polygon.

The remainder of the paper is structured as follows. We begin in Section 2 by giving a simple self-contained proof of our main result for the special case of orthogonal polygons.

We consider generic smooth tracks in Sections 3 and 4. Specifically, in Section 3 we define two important diagrams, which we call the *attraction diagram* and the *dual attraction diagram*, and we prove several useful structural results. At a high level, the attraction diagram is a decomposition of the configuration space $S^1 \times S^1$ according to the puppy's behavior, similar to the *free space diagrams* introduced by Alt and Godau to compute Fréchet distance [5]. We show that for a sufficiently generic smooth track, the attraction diagram consists of a finite number of disjoint simple closed *critical* curves, exactly two of which are topologically nontrivial. Then in Section 4, we argue that the human can catch the puppy on any track whose attraction diagram has this structure.

In Section 5, we sketch an extension of our analysis from smooth curves to simple polygonal tracks; complete details of this extension can be found in the full version of this paper [2]. Because polygons do not have well-defined tangent directions at their vertices, this extension requires explicitly modeling the puppy's direction of motion in addition to its location. We first prove that the human can catch the puppy on a polygon that has no acute vertex angles and where no three vertices form a right angle; under these conditions, the attraction diagram has exactly the same structure as for generic smooth curves. We then reduce the problem for arbitrary simple polygons to this special case by *chamfering* – cutting off a small triangle at each vertex – and arguing that any strategy for catching the puppy on the chamfered track can be pulled back to the original polygon.

Finally, we close the paper by suggesting several directions for further research.

2 Warmup: Orthogonal polygons

As a warmup, we give a simple self-contained proof of our main result for the special case where the track is a simple orthogonal polygon.

▶ **Theorem 1.** *The human can catch the puppy on any simple orthogonal polygon, by walking counterclockwise around the polygon at most twice.*

Proof. Let P be an arbitrary simple orthogonal polygon. Let u_1 be its leftmost point with the maximum y-coordinate, and u_2 be the next boundary vertex of P in clockwise order (see Figure 3). Finally, let ℓ be the horizontal line supporting the segment u_1u_2.

We break the motion of the human into two phases. In the first phase, the human moves counterclockwise around P from their starting location to u_1. If the human catches the puppy during this phase, we are done, so assume otherwise. In the second phase, the human walks counterclockwise around P starting from u_1 to u_2.

We claim that the puppy p is never in the interior of the segment u_1u_2 during the second phase; thus, p always lies on the closed counterclockwise subpath of P from h to u_2 (or less formally, "between h and u_2"). This claim implies that the human and the puppy meet during the second phase on u_2 at the latest.

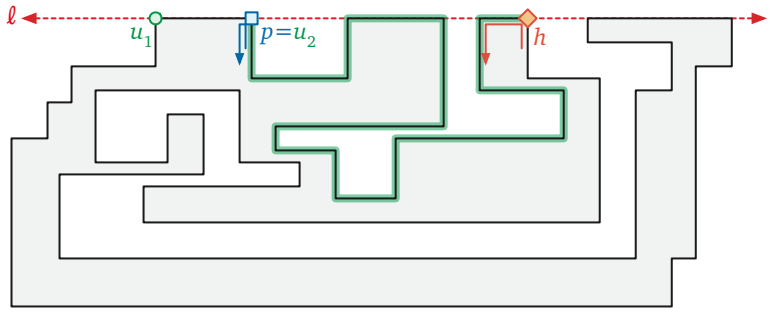


Figure 3 Proof of Theorem 1. During the human's second trip around P, the puppy lies between u_2 and the human.

The puppy must first be at u_2 if it ever wants to be in the interior of u_1u_2. So consider any moment during the second phase when p moves upward to the vertex u_2. At that moment, h must be on the line ℓ to the right of p. (For any point a below ℓ, there is a point b below u_2 that is closer to a than u_2.) Thus, the puppy will stay on u_2 as long as h is on ℓ. As soon as h leaves ℓ the puppy will leave u_2 downward. Thus the puppy can never go to the interior of the edge u_1u_2. ◀

The star-shaped track in Figure 2 shows that this simple argument does not extend to arbitrary polygons, even with a constant number of edge directions.

3 Diagrams of smooth tracks

We first formalize both the problem and our solution under the assumption that the track is a generic smooth simple closed curve $\gamma\colon S^1 \hookrightarrow \mathbb{R}^2$. In particular, for ease of exposition, we assume that γ is regular and C^3, meaning it has well-defined continuous first, second, and third derivatives, and its first derivative is nowhere zero. We also assume γ satisfies some additional genericity constraints, to be specified later. We consider polygonal tracks in Section 5.

3.1 Configurations and genericity assumptions

We analyze the behavior of the puppy in terms of the *configuration space* $S^1 \times S^1$, which is the standard torus. Each configuration point $(x, y) \in S^1 \times S^1$ corresponds to the human being located at $h = \gamma(x)$ and the puppy being located at $p = \gamma(y)$.

For any configuration (x, y), recall that $D(x, y)$ denotes the straight-line Euclidean distance between the points $\gamma(x)$ and $\gamma(y)$. We classify all configurations $(x, y) \in S^1 \times S^1$ into three types, according to the sign of the partial derivative of distance with respect to the puppy's position.

- (x, y) is a *forward* configuration if $\frac{\partial}{\partial y} D(x, y) < 0$.
- (x, y) is a *backward* configuration if $\frac{\partial}{\partial y} D(x, y) > 0$.
- (x, y) is a *critical* configuration if $\frac{\partial}{\partial y} D(x, y) = 0$.

Starting in any forward (resp. backward) configuration, the puppy automatically runs forward (resp. backward) along the track γ. Genericity implies that there are a finite number of critical configurations (x, y) with any fixed value of x, or with any fixed value of y. We further classify the critical configurations as follows:

- (x, y) is a *stable* critical configuration if $\frac{\partial^2}{\partial y^2} D(x, y) > 0$.
- (x, y) is an *unstable* critical configuration if $\frac{\partial^2}{\partial y^2} D(x, y) < 0$.
- (x, y) is a *forward pivot* configuration if $\frac{\partial^2}{\partial y^2} D(x, y) = 0$ and $\frac{\partial^3}{\partial y^3} D(x, y) < 0$.
- (x, y) is a *backward pivot* configuration if $\frac{\partial^2}{\partial y^2} D(x, y) = 0$ and $\frac{\partial^3}{\partial y^3} D(x, y) > 0$.

In any stable configuration, the puppy's distance to the human is locally minimized, so the puppy does not move unless the human moves. In any unstable configuration, the puppy can decrease its distance by running in either direction. Finally, in any forward (resp. backward) pivot configuration, the puppy can decrease its distance by moving in one direction but not the other, and thus automatically runs forward (resp. backward) along the track.

Critical points can also be characterized geometrically as follows. Refer to Figure 4. A configuration (x, y) is critical if the human $\gamma(x)$ lies on the line $N(y)$ normal to γ at the puppy's location $\gamma(y)$. Let $C(y)$ denote the center of curvature of the track at $\gamma(y)$. Then (x, y) is a pivot configuration if $\gamma(x) = C(y)$, a stable critical configuration if the open ray from $C(y)$ through the human point $\gamma(x)$ contains the puppy point $\gamma(y)$, and an unstable critical configuration otherwise.

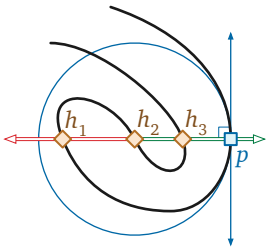


Figure 4 Three critical configurations: (h_1, p) is unstable; (h_2, p) is a pivot configuration, and (h_3, p) is stable.

Genericity of the track γ implies that this classification of critical configurations is exhaustive, and moreover, that the set of pivot configurations is finite. In particular, our analysis requires that in any pivot configuration (x, y), the puppy point $\gamma(y)$ is not a

local curvature minimum or maximum.[1] Otherwise, we would need higher derivatives to disambiguate the puppy's behavior. In the extreme case where γ contains both an open circular arc α and its center c, all configurations where $h = c$ and $p \in \alpha$ are stable.

3.2 Attraction diagrams

The **attraction diagram** of the track γ is a decomposition of the configuration space $S^1 \times S^1$ by critical configurations. Our genericity assumptions imply that the set of critical points – the common boundary of the forward and backward configurations – is the union of a finite number of disjoint simple closed curves, which we call *critical cycles*. At least one of these critical cycles, the main diagonal $x = y$, consists entirely of stable configurations; critical cycles can also consist entirely of unstable configurations. If a critical cycle is neither entirely stable nor entirely unstable, then its points of vertical tangency are pivot configurations, and these points subdivide the curve into x-monotone paths, which alternately consist of stable and unstable configurations.

Figure 5 shows a sketch of the attraction diagram of a simple closed curve. We visualize the configuration torus $S^1 \times S^1$ as a square with opposite sides identified. Green and red paths indicate stable and unstable configurations, respectively; blue dots indicate pivot configurations; and backward configurations are shaded light gray. Figure 6 shows the attraction diagram for a more complex polygonal track, with slightly different coloring conventions. (Again, we will discuss polygonal tracks in more detail in Section 5.)

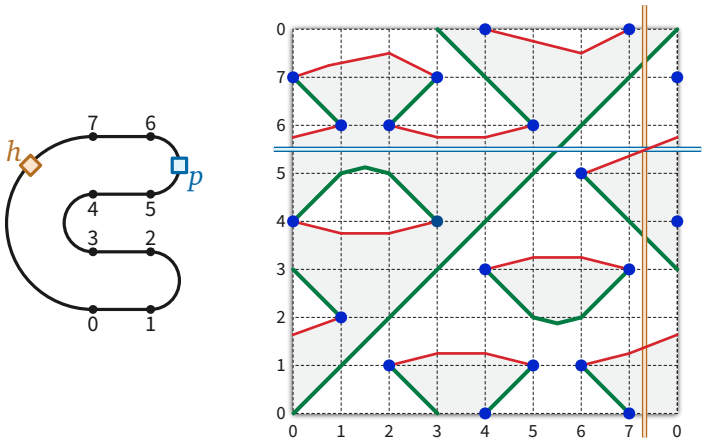


Figure 5 The attraction diagram of a simple closed curve, with one unstable critical configuration emphasized.

The cycles in any attraction diagram have a simple but important topological structure. A critical cycle in the attraction diagram is *contractible* if it is the boundary of a simply connected subset of the torus $S^1 \times S^1$ and *essential* otherwise. For example, the main diagonal is essential, and the attraction diagram in Figure 5 contains two contractible critical cycles and two essential critical cycles.

▶ **Lemma 2.** *The attraction diagram of any generic closed curve contains an even number of essential critical cycles.*

[1] More concretely, we assume the track γ intersects its evolute (the locus of centers of curvature) transversely, away from its cusps.

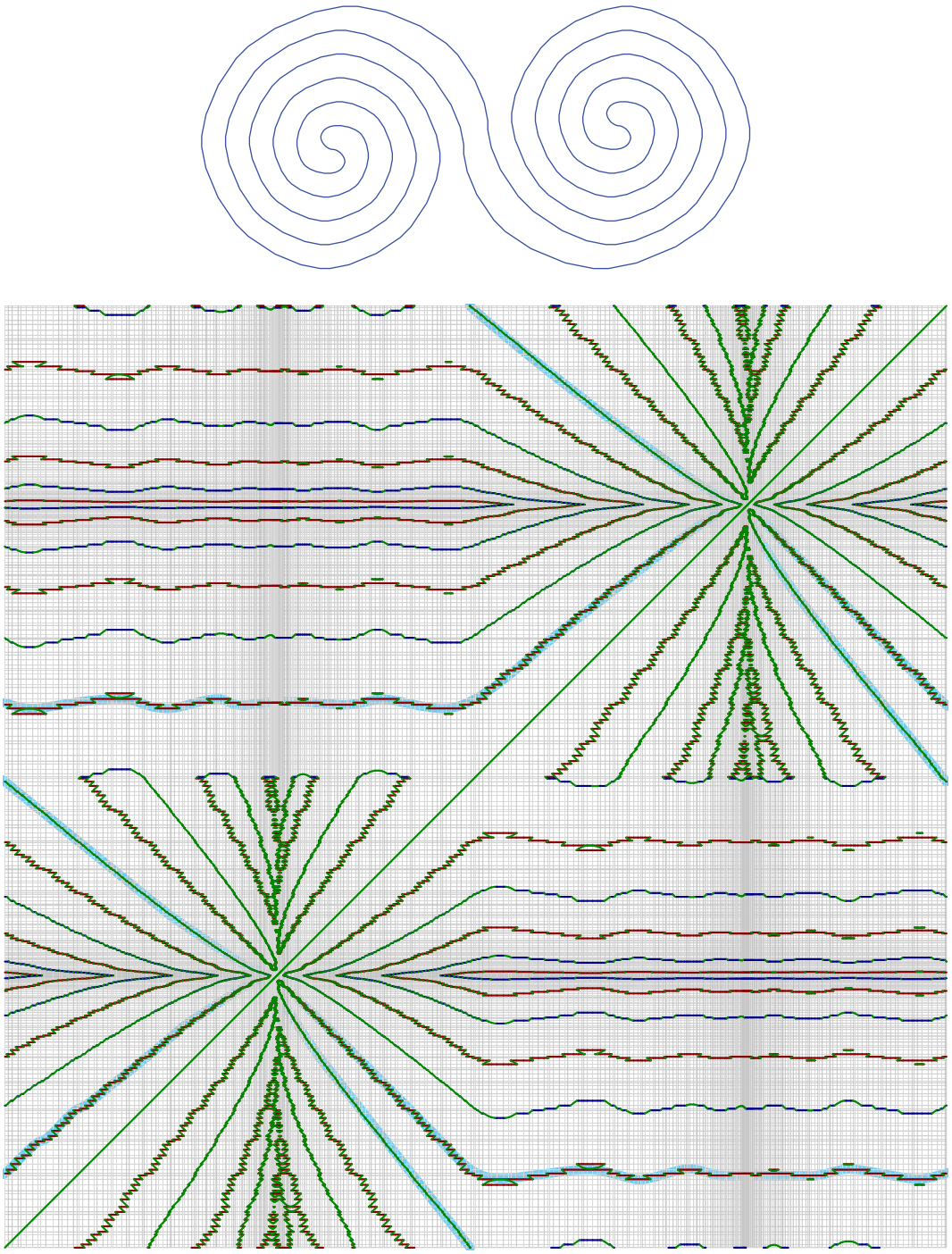


Figure 6 The attraction diagram of a complex simple polygon. Serrations in the diagram are artifacts of the curve being polygonal instead of smooth. The river is highlighted in blue.

Proof. This lemma follows immediately from standard homological arguments, but for the sake of completeness we sketch a self-contained proof.

Fix a generic closed curve γ. Let α be the horizontal cycle $\{(0, y) \mid y \in S^1\}$, and let β be the vertical cycle $\{(x, 0) \mid x \in S^1\}$ in the torus $S^1 \times S^1$. Without loss of generality, assume α and β intersect every critical cycle in the attraction diagram of γ transversely.

A critical cycle C in the attraction diagram is contractible if and only if α and β each cross C an even number of times. (Indeed, this parity condition characterizes all simple contractible closed curves in the torus.) On the other hand, α and β each cross the main diagonal once. It follows that α and β each cross *every* essential critical cycle an odd number of times; otherwise, some pair of essential critical cycles would intersect.

Because the critical cycles are the boundary between the forward and backward configurations, α and β each contain an even number of critical points. The lemma now follows immediately. ◀

We emphasize that this lemma does *not* actually require the track γ to be simple; the argument relies only on properties of generic functions over the torus that are minimized along the main diagonal.

3.3 Dual attraction diagrams

Our analysis also relies on a second diagram, which we call the **dual attraction diagram** of the track. We hope the following intuition is helpful. While the attraction diagram tells us the possible positions of the puppy depending on the position of the human, the dual attraction diagram gives us the possible positions of the human depending on the position of the puppy. For each puppy configuration $y \in S^1$, we consider the normal line $N(y)$. We are interested in the intersection points of γ with $N(y)$, as those are the possible positions of the human. The idea of the dual attraction diagram is to trace the positions of the human as a function of the position of the puppy, see Figure 8.

Let $T(y)$ denote the directed line tangent to γ at the point $\gamma(y)$. For any configuration (x, y), let $\boldsymbol{\ell(x, y)}$ denote the distance from $\gamma(x)$ to the tangent line $T(y)$, signed so that $\ell(x, y) > 0$ if the human point $\gamma(x)$ lies to the left of $T(y)$ and $\ell(x, y) < 0$ if $\gamma(y)$ lies to the right of $T(y)$. More concisely, assuming without loss of generality that the track γ is parameterized by arc length, $\ell(x, y)$ is twice the signed area of the triangle with vertices $\gamma(x)$, $\gamma(y)$, and $\gamma(y) + \gamma'(y)$.

Let $L \colon S^1 \times S^1 \to S^1 \times \mathbb{R}$ denote the function $L(x, y) = (y, \ell(x, y))$. The dual attraction diagram is the decomposition of the infinite cylinder $S^1 \times \mathbb{R}$ by the points $\{L(x, y) \mid (x, y) \text{ is critical}\}$. At the risk of confusing the reader, we refer to the image $L(x, y) \in S^1 \times \mathbb{R}$ of any critical configuration (x, y) as a critical point of the dual attraction diagram.

The dual attraction diagram can also be described as follows. For any $y \in S^1$ and $d \in \mathbb{R}$, let $\Gamma(y, d)$ denote the point on the normal line $N(y)$ at distance d to the left of the tangent vector $\gamma'(y)$. More formally, assuming without loss of generality that γ is parametrized by arc length, we have $\Gamma(y, d) = \gamma(y) + d \left[\begin{smallmatrix} 0 & -1 \\ 1 & 0 \end{smallmatrix}\right] \gamma'(y)$. We emphasize that $\Gamma(y, d)$ does not necessarily lie on the curve γ. The dual attraction diagram is the decomposition of the cylinder $S^1 \times \mathbb{R}$ by the preimage $\Gamma^{-1}(\gamma)$ of γ.

Because γ is simple and regular, the dual attraction diagram is the union of simple disjoint closed curves. The function L continuously maps each critical cycle in the attraction diagram to a closed curve in the cylinder $S^1 \times \mathbb{R}$. Thus, the restriction of L to the set of critical configuration is a homeomorphism onto its image in the dual attraction diagram. In particular, L maps the main diagonal $x = y$ to the horizontal axis $\ell(x, y) = 0$ of the dual

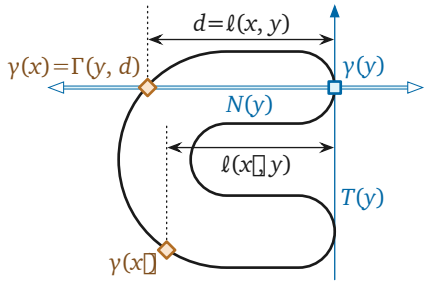


Figure 7 Examples of the functions ℓ and Γ used to define the dual attraction diagram.

attraction diagram. We emphasize, however, that the two diagrams are not topologically equivalent. Figure 8 shows the dual attraction diagram of the same track whose attraction diagram is shown in Figure 5; here preimages of points inside the track are shaded.

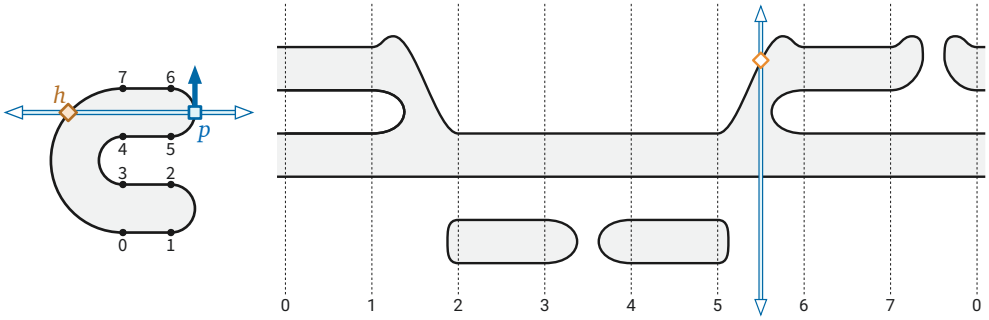


Figure 8 The dual attraction diagram of a simple closed curve, with one critical configuration emphasized. Compare with Figure 5.

▶ **Lemma 3.** *For any generic simple closed curve γ, the attraction diagram of γ and the dual attraction diagram of γ contain the same number of essential critical cycles.*

Proof. Let α denote the horizontal cycle $y = 0$ in the torus $S^1 \times S^1$, and let α' be the vertical line $y = 0$ in the infinite cylinder $S^1 \times \mathbb{R}$. Let C be any critical cycle on the attraction diagram, and let $C' = L(C)$ be the corresponding critical cycle in the dual attraction diagram.

Recall from the proof of Lemma 2 that C is contractible on the torus if and only if $|C \cap \alpha|$ is even. Similarly, C' is contractible in the cylinder if and only if $|C' \cap \alpha'|$ is even. The map $L\colon S^1 \times S^1 \to S^1 \times \mathbb{R}$ maps $C \cap \alpha$ bijectively to $C' \cap \alpha'$. We conclude that C is essential if and only if C' is essential. ◀

With this correspondence in hand, we can now more carefully describe the topological structure of the *attraction* diagram when the track is simple.

▶ **Lemma 4.** *The attraction diagram of a **simple** generic closed curve contains **two** essential critical cycles.*

Proof. Fix a generic closed curve γ. Lemma 2 implies that the attraction diagram of γ contains at least two essential critical cycles, one of which is the main diagonal. Thus, to prove the lemma, it remains to show that there are *at most* two essential critical cycles, in either the attraction diagram or the dual attraction diagram.

Let $\Sigma \subset S^1 \times \mathbb{R}$ denote the set of essential critical cycles in the *dual attraction* diagram. Any two cycles in Σ are homotopic – meaning one can be continuously deformed into the other – because there is only one nontrivial homotopy class of simple cycles on the infinite cylinder $S^1 \times \mathbb{R}$. It follows that the cycles in Σ have a well-defined vertical total order. In particular, the highest and lowest intersection points between any vertical line and Σ always lie on the *same* two essential cycles in Σ.

Without loss of generality, suppose $\gamma(0)$ is a point on the convex hull of γ with a unique tangent line. Let C be any essential critical cycle in the attraction diagram of γ, and let $C' = L(C)$ denote the corresponding essential cycle in the dual attraction diagram. C must pass through all possible puppy positions *and* all possible human positions; thus, C contains a configuration $(0, y)$ for some parameter $y \in S^1$. Recall that $N(y)$ denotes the line normal to γ at $\gamma(y)$. Then $\gamma(0)$ must also lie on the convex hull of $\gamma \cap N(y)$. We conclude that C' must be either the highest or lowest essential critical cycle in the dual attraction diagram. We conclude that there are at most two critical cycles, completing the proof. ◀

In the rest of the paper, we mnemonically refer to the two essential critical cycles in the attraction diagram of a simple track as the **main diagonal** and the **river**.

We emphasize that the converse of Lemma 4 is false; there are non-simple tracks whose attraction diagrams have exactly two essential critical cycles. (Consider the figure-eight curve ∞.) Moreover, we conjecture that Lemma 4 can be generalized to all (smooth) tracks with turning number ± 1.

4 Dexter and sinister strategies

We can visualize any strategy for the human to catch the puppy as a path through the attraction diagram that consists entirely of segments of stable critical paths and vertical segments, as shown in Figure 9. We refer to the vertical segments as *pivots*. Every pivot (except possibly the first) starts at a pivot configuration, and every pivot ends at a stable configuration.

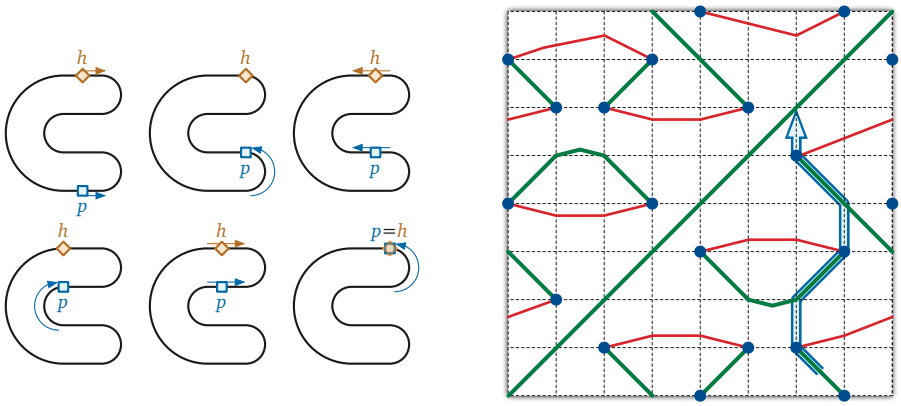


Figure 9 A sinister strategy for catching the puppy; compare with Figures 1 and 5.

We call a strategy **dexter** if it ends with a backward pivot – a *downward* segment, approaching the main diagonal to the *right* – and we call a configuration (x, y) *dexter* if there is a dexter strategy for catching the puppy starting at (x, y). Similarly, a strategy is

sinister if it ends with a forward pivot – a *skyward* segment, approaching the main diagonal to the *left* – and a configuration is sinister if it is the start of a sinister strategy.[2] A single configuration can be both dexter and sinister; see Figure 10.

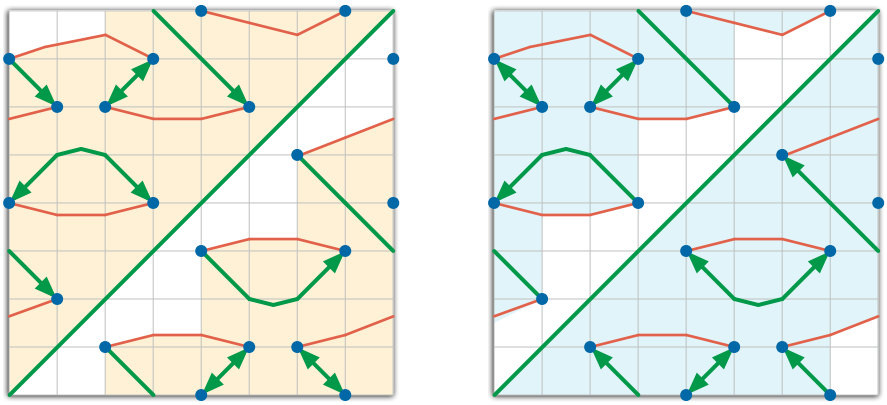


Figure 10 Dexter (orange) and sinister (cyan) configurations in the example attraction diagram. Arrows on the stable critical paths describe dexter and sinister strategies for catching the puppy.

▶ **Theorem 5.** *Let γ be a generic track whose attraction diagram has exactly two essential critical cycles. Every configuration on γ is either dexter or sinister; thus, the human can catch the puppy on γ from any starting configuration.*

Before giving the proof, we emphasize that Theorem 5 does not require the track γ to be simple. Also, it is an open question whether having exactly two essential critical cycle curves is a *necessary* condition for the human to always be able to catch the puppy. (We conjecture that it is not.)

Proof. Fix a generic track γ whose attraction diagram has exactly two essential critical cycles, which we call the *main diagonal* and the *river*. Assume γ has at least one pivot configuration, since otherwise, from any starting configuration, the puppy runs directly to the human.

Let D be the set of all dexter configurations, and let S be the set of all sinister configurations. We claim that D and S are both annuli that contain both the main diagonal and the river. Because S and D meet on opposite sides of the main diagonal, this claim implies that $D \cup S$ is the entire torus, completing the proof of the lemma. We prove our claim explicitly for D; a symmetric argument establishes the claim for S.

For purposes of argument, we partition the attraction diagram of γ by extending vertical segments from each pivot configuration to the next critical cycles directly above and below. We call the cells in this decomposition *trapezoids*, even though their top and bottom boundaries may not be straight line segments. At each forward pivot configuration p, we color the vertical segment above (x, y) *green* and the vertical segment below p *red*; the colors are reversed for backward vertical segments, see Figure 11.

The first step of any strategy is a (possibly trivial) pivot onto a stable critical path. Because the human and puppy can move freely within any stable critical path σ, either every point in σ is dexter, or no point in σ is dexter. Similarly, for any green pivot segment π, either every point in π is dexter or no point in π is dexter.

[2] *Dexter* and *sinister* are Latin for right (or skillful, or fortunate, or proper, from a Proto-Indo-European root meaning "south") and left (or unlucky, or unfavorable, or malicious), respectively.

Consider any trapezoid τ, and let σ be the stable critical path on its boundary. Starting in any configuration in τ, the puppy immediately moves to a configuration on σ. Thus, if any point in τ is dexter, then σ is dexter, which implies that *every* point in τ is dexter. Thus, we can describe entire trapezoids as dexter or not dexter. It follows that D is the union of trapezoids.

If two trapezoids share a stable critical path *other than the main diagonal*, then either both trapezoids are dexter or neither is dexter. Similarly, if the green pivot segment leaving a pivot configuration p is dexter, then all four trapezoids incident to p are dexter; otherwise, either two or none of these four trapezoids are dexter.

We conclude that aside from the main diagonal, the boundary of D consists entirely of unstable critical paths, pivot configurations, and red vertical segments. Moreover, for every pivot configuration p on the boundary of D, the green pivot segment leaving p is *not* dexter.

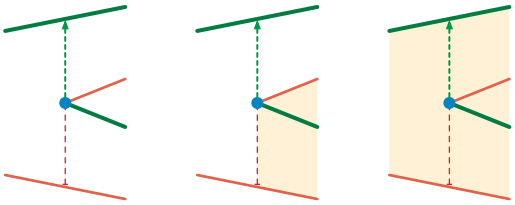


Figure 11 Possible arrangements of dexter trapezoids near a forward pivot configuration.

By definition, every point in D is connected by a (dexter) path to the main diagonal, so D is non-empty and connected. On the other hand, D excludes a complete cycle of forward configurations just below the main diagonal. For any $x \in S^1$, let $D(x)$ denote the set of dexter configurations (x, y); this set consists of one or more vertical line segments in the attraction diagram.

Suppose for the sake of argument that some set $D(x)$ is disconnected. Because D is connected, the boundary of D must contain a *concave vertical bracket*: A vertical boundary segment π whose adjacent critical boundary segments both lie (without loss of generality) to the right of π, but D lies locally to the left of π. See Figure 12. Let p be the pivot configuration at one end of π. The green vertical segment on the other side of p is dexter, which implies that *all* trapezoids incident to p are dexter, contradicting the assumption that π lies on the boundary of D. We conclude that for all x, the set $D(x)$ is a single vertical line segment; in other words, D is a *monotone* annulus.

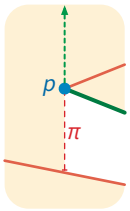


Figure 12 A hypothetical concave vertical bracket on the boundary of D.

The bottom boundary of D is the main diagonal. The monotonicity of D implies that the top boundary of D is a monotone "staircase" alternating between upward red vertical segments and rightward unstable critical paths. Every trapezoid immediately above the top boundary of D contains only forward configurations. Thus, there is a complete essential cycle ϕ of forward configurations just above the upper boundary of D. Because ϕ contains only forward configurations, ϕ must lie entirely above the river. It follows that D contains the entire river.

Symmetrically, S is an annulus bounded above by the main diagonal and bounded below by a non-contractible cycle of backward configurations; in particular, the entire river lies inside S. We conclude that $D \cup S$ is the entire configuration torus. ◀

If the attraction diagram of γ has more than two essential critical cycles curves, then D and S are still monotone annuli, each bounded by the main diagonal and an essential cycle of red vertical segments and unstable paths, and thus S and D each contain at least one essential critical cycle other than the main diagonal. However, $D \cup S$ need not cover the entire torus.

▶ **Corollary 6.** *The human can catch the puppy on any generic simple closed track, from any starting configuration.*

5 Polygonal tracks

Our previous arguments require, at a minimum, that the track has a continuous derivative that is never equal to zero. We now extend our analysis to arbitrary polygonal tracks, which do not have well-defined tangent directions at their vertices. Due to space constraints, we only sketch the main ideas here; a complete discussion can be found in the full version of our paper [2]. Our high-level strategy has two stages. First we argue that suitably defined attraction and dual attraction diagrams for *generic* polygons have the same structural properties as the corresponding diagrams of smooth curves. Then we reduce the analysis of arbitrary polygons to the generic case by *chamfering* – cutting small triangles from all polygon vertices.

To properly describe the puppy's behavior, we must account for the direction that the puppy is facing, even when the puppy lies at a vertex. To that end, we represent the puppy using both a continuous *position* function $\pi\colon S^1 \to \mathbb{R}$ and a continuous *direction* function $\theta\colon S^1 \to S^1$, such that for all $y \in S^1$, the derivative vector $\pi'(y)$ is a non-negative scalar multiple of the unit vector $\theta(y)$. Intuitively, as we increase y, the puppy alternately moves at constant speed along edges, without changing direction (where $\pi'(y) \parallel \theta(y)$ and $\theta'(y) = 0$) and rotates at constant speed at vertices, without changing position (where $\pi'(y) = 0$ and $\theta'(y) \neq 0$).

To define the attraction diagram of P, we decompose the torus $S^1 \times S^1$ into a $2n \times n$ grid of rectangular cells, where each column corresponds to an edge e_j containing the human, and each row corresponds to either a vertex v_i or an edge e_i containing the puppy. See Figure 13 (left) for a complete example.

We now analyze the structure of polygonal attraction diagrams. Each edge-edge cell $e_i \times e_j$ contains at most one boundary-to-boundary path of stable critical configurations (x, y). Refer to Figure 14.

Each vertex-edge cell $v_i \times e_j$ contains at most one boundary-to-boundary path of stable critical configurations and at most one boundary-to-boundary path of unstable critical configurations. A configuration (x, y) with $\pi(y) = v_i$ is stable if and only if $P(x)$ lies in the outer normal cone at v_i, and unstable if and only if $P(x)$ lies in the inner normal cone at v_i. Refer to Figure 15.

Unlike the attraction diagrams of generic smooth curves defined in Section 3.2, attraction diagrams of polygons are not always well-behaved. In particular, a pivot configuration may be incident to more (or fewer) than two critical curves, and in extreme cases, pivot configurations need not even be discrete. We call such a configuration a *degenerate* pivot configuration; see Figure 16.

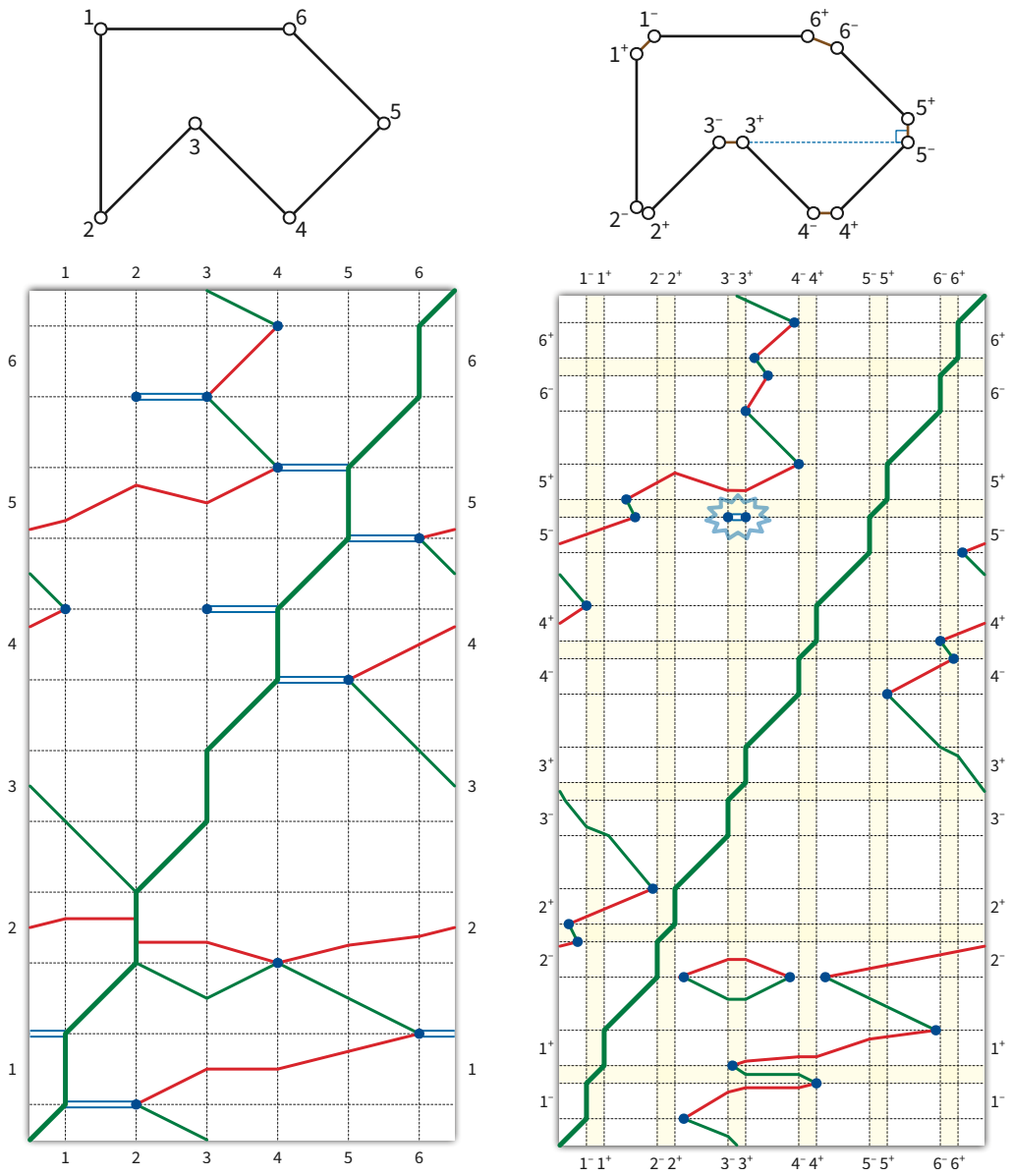


Figure 13 The attraction diagram of a degenerate polygon, before and after chamfering. All existing degeneracies disappeared in the chamfered polygon, which does have one new but harmless degeneracy.

We first consider polygonal tracks which do not have any degeneracies. Generic obtuse polygonal tracks behave almost identically to smooth tracks. In particular, we prove that the polygonal attraction diagram of P is the union of disjoint simple critical cycles, that it contains exactly two essential critical cycles, and that if the attraction diagram of P has exactly two essential critical cycles, then the human can catch the puppy on P, starting from any initial configuration.

Finally, we extend our analysis to arbitrary simple polygons. We define a *chamfering* operation which transforms a polygon P into a new polygon $\bar{P}$, intuitively by cutting off a small triangle at each vertex, as shown in Figure 17. First we show that chamfering a

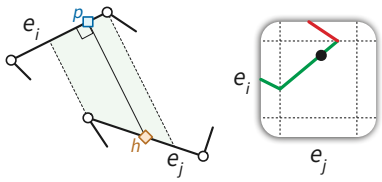


Figure 14 All edge-edge critical configurations are stable.

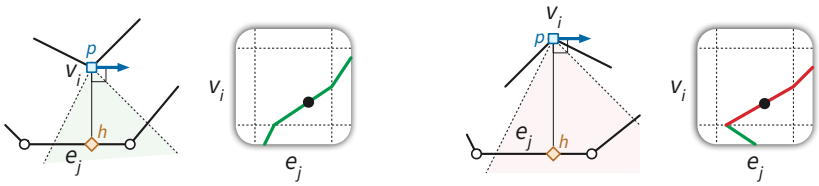


Figure 15 Stable and unstable vertex-edge critical configurations.

polygon removes *most* of its degenerate configurations – see Figure 13 (right) for an example. All remaining degeneracies cause either isolated critical vertices or degenerate pivot edges in the attraction diagram; these remaining degeneracies do not impact the existence of a strategy to catch the puppy on $\bar{P}$. Finally, we argue that any strategy on $\bar{P}$ can be correctly translated back to a strategy on the original polygon P.

6 Further questions

For simple curves, we have only proved that a catching strategy exists. At least for polygonal tracks, it is straightforward to compute such a strategy in $O(n^2)$ time by searching the attraction diagram. In fact, we can compute a strategy that minimizes the total distance traveled by either the human or the puppy in $O(n^2)$ time, using fast algorithms for shortest paths in toroidal graphs [16, 18]. Unfortunately, this quadratic bound is tight in the worst case if the output strategy must be represented as an explicit path through the attraction diagram. We conjecture that an optimal strategy can be described in only $O(n)$ space by listing only the human's initial direction and the sequence of points where the human reverses direction. On the other hand, an algorithm to compute such an optimal strategy in subquadratic time seems unlikely.

If the track is a *smooth curve* of length ℓ whose attraction diagram has k pivot configurations, a trivial upper bound on the distance the human must walk to catch the puppy is $\ell \cdot k/2$. In any optimal strategy, the human walks straight to the point on the curve corresponding to a pivot located at one of the two endpoints of the current "stable sub-curve" of a critical curve (walking less than ℓ). Then the configuration moves to another stable sub-curve, and so on, never visiting the same stable sub-curve twice. Our question is whether a better upper bound can be proved.

In fact, if minimizing distance is not a concern, we conjecture that *no* reversals are necessary. That is, on *any* simple track, starting from *any* configuration, we conjecture that the human can catch the puppy *either* by walking only forward along the track *or* by walking only backward along the track. Figure 2 and its reflection show examples where each of these naïve strategies fails, but we have no examples where both fail. (Our proof of Theorem 1 implies that the human can always catch the puppy on an *orthogonal* polygon by walking *at most once* around the track in some direction, depending on the starting configuration.)

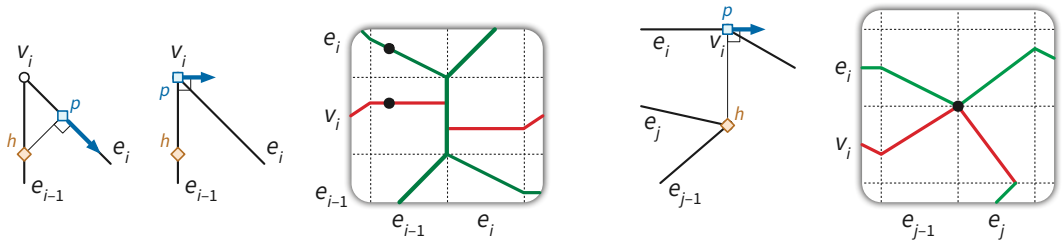


Figure 16 Examples of degenerate pivot configurations, caused by an acute vertex (left) or a vertex lying on a line perpendicular to an edge (right).

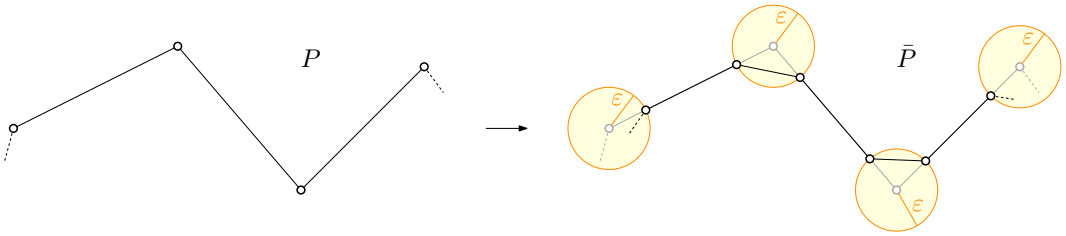


Figure 17 The chamfering operation.

More ambitiously, we conjecture that the following *oblivious* strategy is always successful: walk twice around the track in one (arbitrary) direction, then walk twice around the track in the opposite direction.

Another interesting question is to what extent our result applies to curves in $\mathbb{R}^3$, or to self-intersecting curves in the plane, when we consider the two strands of the curve at an intersection point to be distinct. It is easy to see that the human cannot catch the puppy on a curve that traverses a circle twice; see Figure 18. Indeed, we know how to construct examples of bad curves with any rotation number *except* -1 and $+1$. We conjecture that Lemma 4, and therefore our main result, extends to all non-simple tracks with rotation number ± 1.

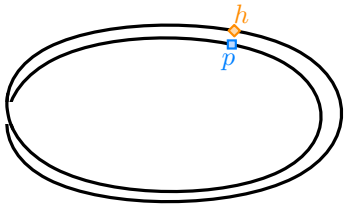


Figure 18 A double loop; p and h will never meet.

Finally, it is natural to consider similar pursuit-attraction problems in more general domains. In the full version of the paper [2], we prove that human can catch the puppy in the interior of any simple polygon by walking along the dual tree of any triangulation. Can the human always catch the puppy in any planar straight-line graph? Inside any polygon with holes?

References

1 Zachary Abel, Hugo A. Akitaya, Erik D. Demaine, Martin L. Demaine, Adam Hesterberg, Matias Korman, Jason S. Ku, and Jayson Lynch. Negative instance for the edge patrolling beacon problem. *Preprint*, 2020. arXiv:2006.01202.

2 Mikkel Abrahamsen, Jeff Erickson, Irina Kostitsyna, Maarten Löffler, Tillmann Miltzow, Jérôme Urhausen, Jordi Vermeulen, and Giovanni Viglietta. Chasing puppies: Mobile beacon routing on closed curves, 2021. arXiv:2103.09811.

3 Israel Aldana-Galván, Jose L. Alvarez-Rebollar, Juan Carlos Cataa-Salazar, Nestaly Marin-Nevárez, Erick Solís-Villareal, Jorge Urrutia, and Carlos Velarde. Beacon coverage in orthogonal polyhedra. In *Proceedings of the 29th Canadian Conference on Computational Geometry (CCCG 2017)*, pages 156–161, 2017.

4 Israel Aldana-Galván, Jose L. Alvarez-Rebollar, Juan Carlos Cataa-Salazar, Nestaly Marin-Nevárez, Erick Solís-Villareal, Jorge Urrutia, and Carlos Velarde. Tight bounds for illuminating and covering of orthotrees with vertex lights and vertex beacons. *Graphs and Combinatorics*, 36:617–630, 2929.

5 Helmut Alt and Michael Godau. Computing the Fréchet distance between two polygonal curves. *Int. J. Comput. Geom. Appl.*, 5:75–91, 1995. doi:10.1142/S0218195995000064.

6 Sang Won Bae, Chan-Su Shin, and Antoine Vigneron. Tight bounds for beacon-based coverage in simple rectilinear polygons. *Computational Geometry*, 80:40–52, 2019. doi:10.1016/j.comgeo.2019.02.002.

7 Michael Biro. *Beacon-based routing and guarding*. PhD thesis, State University of New York at Stony Brook, 2013.

8 Michael Biro, Jie Gao, Justin Iwerks, Irina Kostitsyna, and Joseph S. B. Mitchell. Beacon-based routing and coverage. In *Proceedings of the 21st Fall Workshop on Computational Geometry*, 2011.

9 Michael Biro, Jie Gao, Justin Iwerks, Irina Kostitsyna, and Joseph S. B. Mitchell. Beacon based structures in polygonal domains. In *Abstracts of the 1st Computa- tional Geometry: Young Researchers Forum*, 2012.

10 Michael Biro, Jie Gao, Justin Iwerks, Irina Kostitsyna, and Joseph S. B. Mitchell. Combinatorics of beacon routing and coverage. In *Proceedings of the 25th Canadian Conference on Computational Geometry (CCCG 2013)*, 2013.

11 Michael Biro, Justin Iwerks, Irina Kostitsyna, and Joseph S. B. Mitchell. Beacon-based algorithms for geometric routing. In *13th International Symposium on Algorithms and Data Structures (WADS 2013)*, pages 158–169, 2013. doi:10.1007/978-3-642-40104-6_14.

12 Prosenjit Bose and Thomas C. Shermer. Gathering by repulsion. *Computational Geometry*, 90:101627, 2020. doi:10.1016/j.comgeo.2020.101627.

13 Pierre Bouguer. Sur de nouvelles courbes ausquelle on peut donner le nom de linges de poursuite. *Mémoires de mathématique et de physique tirés des registres de l'Académie royale des sciences*, pages 1–14, 1732. URL: https://gallica.bnf.fr/ark:/12148/bpt6k35294.

14 Johans Cleve and Wolfgang Mulzer. Combinatorics of beacon-based routing in three dimensions. *Computational Geometry*, 91:101667, 2020. doi:10.1016/j.comgeo.2020.101667.

15 Pierre de Maupertuis. Sure les courbes de poursuite. *Mémoires de mathématique et de physique tirés des registres de l'Académie royale des sciences*, pages 15–17, 1732. URL: https://gallica.bnf.fr/ark:/12148/bpt6k35294.

16 John R. Gilbert, Joan P. Hutchinson, and Robert Endre Tarjan. A separator theorem for graphs of bounded genus. *J. Algorithms*, 5(3):391–407, 1984. doi:10.1016/0196-6774(84)90019-1.

17 Arthur S. Hathaway. Problems and solutions: Problem 2801. *American Mathematical Monthly*, 27(1):31, 1920. doi:10.2307/2973244.

18 Monika R. Henzinger, Philip Klein, Satish Rao, and Sairam Subramanian. Faster shortest-path algorithms for planar graphs. *J. Comput. Syst. Sci.*, 55(1):3–23, 1997. doi:10.1006/jcss.1997.1493.

19 Irina Kostitsyna, Bahram Kouhestani, Stefan Langerman, and David Rappaport. An Optimal Algorithm to Compute the Inverse Beacon Attraction Region. In *34th International Symposium on Computational Geometry (SoCG 2018)*, 2018. doi:10.4230/LIPIcs.SoCG.2018.55.

20 Bahram Kouhestani and David Rappaport. Edge patrolling beacon. In *Abstracts from the 20th Japan Conference on Discrete and Computational Geometry, Graphs, and Games (JCDCGGG 2017)*, pages 101–102, 2017.

21 Bahram Kouhestani, David Rappaport, and Kai Salomaa. On the inverse beacon attraction region of a point. In *Proceedings of the 27th Canadian Conference on Computational Geometry (CCCG 2015)*, 2015.

22 Bahram Kouhestani, David Rappaport, and Kai Salomaa. The length of the beacon attraction trajectory. In *Proceedings of the 28th Canadian Conference on Computational Geometry (CCCG 2016)*, pages 69–74, 2016.

23 Bahram Kouhestani, David Rappaport, and Kai Salomaa. Routing in a polygonal terrain with the shortest beacon watchtower. *Computational Geometry*, 68:34–47, 2018. doi:10.1016/j.comgeo.2017.05.005.

24 John E. Littlewood. *Littlewood's Miscallany: edited by Béla Bollobás*. Cambridge University Press, 1986.

25 Amirhossein Mozafari and Thomas C. Shermer. Transmitting particles in a polygonal domain by repulsion. In *12th International Conference on Combinatorial Optimization and Applications (COCOA 2018)*, pages 495–508, 2018. doi:10.1007/978-3-030-04651-4_33.

26 Paul J. Nahin. *Chases and Escapes: The Mathematics of Pursuit and Evasion*. Princeton University Press, 2007.

27 Thomas C. Shermer. A combinatorial bound for beacon-based routing in orthogonal polygons. In *Proceedings of the 27th Canadian Conference on Computational Geometry (CCCG 2015)*, 2015.

Online Packing to Minimize Area or Perimeter

Mikkel Abrahamsen
BARC, University of Copenhagen, Denmark

Lorenzo Beretta
BARC, University of Copenhagen, Denmark

Abstract

We consider online packing problems where we get a stream of axis-parallel rectangles. The rectangles have to be placed in the plane without overlapping, and each rectangle must be placed without knowing the subsequent rectangles. The goal is to minimize the perimeter or the area of the axis-parallel bounding box of the rectangles. We either allow rotations by 90° or translations only.

For the perimeter version we give algorithms with an absolute competitive ratio slightly less than 4 when only translations are allowed and when rotations are also allowed.

We then turn our attention to minimizing the area and show that the competitive ratio of any algorithm is at least $\Omega(\sqrt{n})$, where n is the number of rectangles in the stream, and this holds with and without rotations. We then present algorithms that match this bound in both cases and the competitive ratio is thus optimal to within a constant factor. We also show that the competitive ratio cannot be bounded as a function of Opt. We then consider two special cases.

The first is when all the given rectangles have aspect ratios bounded by some constant. The particular variant where all the rectangles are squares and we want to minimize the area of the bounding square has been studied before and an algorithm with a competitive ratio of 8 has been given [Fekete and Hoffmann, Algorithmica, 2017]. We improve the analysis of the algorithm and show that the ratio is at most 6, which is tight.

The second special case is when all edges have length at least 1. Here, the $\Omega(\sqrt{n})$ lower bound still holds, and we turn our attention to lower bounds depending on Opt. We show that any algorithm for the translational case has a competitive ratio of at least $\Omega(\sqrt{\text{Opt}})$. If rotations are allowed, we show a lower bound of $\Omega(\sqrt[4]{\text{Opt}})$. For both versions, we give algorithms that match the respective lower bounds: With translations only, this is just the algorithm from the general case with competitive ratio $O(\sqrt{n}) = O(\sqrt{\text{Opt}})$. If rotations are allowed, we give an algorithm with competitive ratio $O(\min\{\sqrt{n}, \sqrt[4]{\text{Opt}}\})$, thus matching both lower bounds simultaneously.

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Packing and covering problems; Theory of computation → Online algorithms

Keywords and phrases Packing, online algorithms

Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.6

Related Version *Full Version*: https://arxiv.org/abs/2101.09024

Funding Research of both authors partly supported by Investigator Grant 16582, Basic Algorithms Research Copenhagen (BARC), from the VILLUM Foundation.
Lorenzo Beretta: receives funding from the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No 801199.

1 Introduction

Problems related to packing appear in a plethora of big industries. For instance, two-dimensional versions of packing arise when a given set of pieces have to be cut out from a large piece of material so as to minimize waste. This is relevant to clothing production where cutting patterns are cut out from a roll of fabric, and similarly in leather, glass, wood, and sheet metal cutting.

In some applications, it is important that the pieces are placed in an *online* fashion. This means that the pieces arrive one by one and we need to decide the placement of one piece before we know the ones that will come in the future. This is in contrast to *offline* problems, where all the pieces are known in advance. Problems related to packing were some of the first for which online algorithms were described and analyzed. Indeed, the first use of the terms "online" and "offline" in the context of approximation algorithms was in the early 1970s and used for algorithms for bin-packing problems [14].

In this paper, we study online packing problems where the pieces can be placed anywhere in the plane as long as they do not overlap. The goal is to minimize the region occupied by the pieces. The pieces are axis-parallel rectangles, and they may or may not be rotated by 90°. We want to minimize the size of the axis-parallel bounding box of the pieces, and the size of the box is either the perimeter or the area. This results in four problems: PerimeterRotation, PerimeterTranslation, AreaRotation, and AreaTranslation.

Competitive analysis

The *competitive ratio* of an online algorithm is the equivalent of the *approximation ratio* of an (offline) approximation algorithm. The usual definitions [7, 9, 11] of competitive ratio (or *worst case ratio*, as it may also be called [11]) can only be used to describe that the cost of the solution produced by an online algorithm is at most some constant factor higher than the cost Opt of the optimal (offline) solution. In the study of approximation algorithms, it is often the case that the approximation ratio is described not just as a constant, but as a more general function of the input. In the same way, we generalize the definition of competitive ratios to support such statements about online algorithms.

Consider an algorithm A for one of the packing problems studied in this paper. Let $\mathcal{L}$ be the set of non-empty streams of rectangular pieces. For a stream $L \in \mathcal{L}$, we define $A(L)$ to be the cost of the packing produced by A and let $\textsc{Opt}(L)$ be the cost of the optimal (offline) packing. We say that A has a *competitive ratio* of $f(L)$, for some function $f : \mathcal{L} \longrightarrow \mathbb{R}^+$ which may just be a constant, if

$$\sup_{L \in \mathcal{L}} \frac{A(L)}{\textsc{Opt}(L) f(L)} \leq 1.$$

In this paper, the functions $f(L)$ that we consider will be (i) constants, (ii) functions of the number of pieces $n = |L|$, (iii) functions of $\textsc{Opt}(L)$.

Results and structure of the paper

We develop online algorithms for the perimeter versions PerimeterRotation and PerimeterTranslation, both with a competitive ratio slightly less than 4. These algorithms are described in Section 2. The idea is to partition the positive quadrant into *bricks*, which are axis-parallel rectangles with aspect ratio $\sqrt{2}$. In each brick, we build a stack of pieces which would be too large to place in a brick of smaller size. Online packing algorithms using higher-dimensional bricks were described by Januszewski and Lassak [15] and our algorithms are inspired by an algorithm of Fekete and Hoffmann [13].

In Section 3, we study the area versions AreaRotation and AreaTranslation. We show in Section 3.1 that any algorithm A processing a stream of n pieces cannot achieve a better competitive ratio than $\Omega(\sqrt{n})$, and this holds for all online algorithms and with and without rotations allowed. It also holds in the special case where all the edges of pieces have length at least 1. We furthermore show that when the pieces can be arbitrary, no bound on

the competitive ratio as a function of Opt for AreaRotation nor AreaTranslation. In Section 3.2 we describe the algorithms DynBoxTrans and DynBoxRot, which achieve a $O(\sqrt{n})$ competitive ratio for AreaTranslation and AreaRotation, respectively, for an arbitrary stream of n pieces. This is thus optimal up to a constant factor when measuring the competitive ratio as a function of n. Both algorithms use a row of boxes of exponentially increasing width and dynamically adjusted height. In these boxes, we pack pieces using a next-fit shelf algorithm, which is a classic online strip packing algorithm first described by Baker and Schwartz [6].

We then turn our attention to two special cases. The first special case is when the aspect ratio is bounded by a constant $\alpha \geq 1$. A case of particular interest is when all pieces are squares, i.e., $\alpha = 1$. It is natural to have the same requirement to the container as to the pieces, so let us assume that the goal is to minimize the area of the axis-parallel bounding square of the pieces, and call the problem SquareInSquareArea. This problem was studied by Fekete and Hoffmann [13], and they gave an algorithm for the problem and proved that it was 8-competitive. We prove that the same algorithm is in fact 6-competitive and that this is tight. It easily follows that if the aspect ratio is bounded by an arbitrary constant $\alpha \geq 1$ or if the goal is to minimize the area of the axis-parallel bounding rectangle, we also get a $O(1)$-competitive algorithm.

The second special case is when all edges are *long*, that is, when they have length at least 1 (any other constant will work too). In Section 3.4, we show that under this assumption, there is a lower bound of $\Omega(\sqrt{\text{Opt}})$ for the competitive ratio of AreaTranslation, whereas for AreaRotation, we get the lower bound $\Omega(\sqrt[4]{\text{Opt}})$. In Section 3.5, we provide algorithms for the area versions when the edges are long. For both problems AreaRotation and AreaTranslation, we give algorithms that match the lower bounds of Section 3.4 to within a constant factor. With translations only, this is just the algorithm from the general case with competitive ratio $O(\sqrt{n}) = O(\sqrt{\text{Opt}})$. The algorithm with ratio $O(\sqrt[4]{\text{Opt}})$ for the rotational case follows the same scheme as the algorithms for arbitrary rectangles of Section 3.2, but differ in the way we dynamically increase boxes' heights. We finally describe an algorithm for the rotational case with competitive ratio $O(\min\{\sqrt{n}, \sqrt[4]{\text{Opt}}\})$, thus matching the lower bounds $\Omega(\sqrt{n})$ and $\Omega(\sqrt[4]{\text{Opt}})$ simultaneously. Actually, the two lower bounds for AreaRotation can be summarized by $\Omega(\max\{\sqrt{n}, \sqrt[4]{\text{Opt}}\})$, while we manage to achieve a competitive ratio of $O(\min\{\sqrt{n}, \sqrt[4]{\text{Opt}}\})$. However, this gives no contradiction, it simply proves that the *edge cases* that have a competitive ratio of at least $\Omega(\sqrt[4]{\text{Opt}})$ must satisfy $\text{Opt} = O(n^2)$, and those for which the competitive ratio is at least $\Omega(\sqrt{n})$ satisfy $n = O(\sqrt{\text{Opt}})$. We summarize the results in Table 1.

Related work

The literature on online packing problems is rich. See the surveys of Christensen, Khan, Pokutta, and Tetali [9], van Stee [25, 26], and Csirik and Woeginger [11] for an overview. It seems that the vast majority of previous work on online versions of two-dimensional packing problems is concerned with either bin packing (packing the pieces into a minimum number of unit squares) or strip packing (packing the pieces into a strip of unit width so as to minimize the total height of the pieces). From a mathematical point of view, we find the problems studied in this paper perhaps even more fundamental than these important problems in the sense that we give no restrictions on where to place the pieces, whereas the pieces are restricted by the boundaries of the bins and the strip in bin and strip packing.

Table 1 Results of this paper.

Measure	Version	Trans./Rot.	Lower bound	Upper bound
Perimeter	General	Translation	4/3, Sec. 2.2	$4-\varepsilon$, Sec. 2.1
		Rotation	5/4, Sec. 2.2	$4-\varepsilon$, Sec. 2.1
Area	General	Translation	$\Omega(\sqrt{n})$ & $\forall f : \Omega(f(\textsc{Opt}))$, Sec. 3.1	$O(\sqrt{n})$, Sec. 3.2
		Rotation	$\Omega(\sqrt{n})$ & $\forall f : \Omega(f(\textsc{Opt}))$, Sec. 3.1	$O(\sqrt{n})$, Sec. 3.2
	Sq.-in-sq.	N/A	16/9, Sec. 3.3	6, Sec. 3.3
	Long edges	Translation	$\Omega(\sqrt{\textsc{Opt}})$, Sec. 3.4	$O(\sqrt{n}) = O(\sqrt{\textsc{Opt}})$, Sec. 3.5
		Rotation	$\Omega(\max\{\sqrt{n}, \sqrt[4]{\textsc{Opt}}\})$, Sec. 3.1 and 3.4	$O(\min\{\sqrt{n}, \sqrt[4]{\textsc{Opt}}\})$, Sec. 3.5

Another related problem is to find the critical density of online packing squares into a square. In other words, what is the maximum $\Sigma \leq 1$ such that there is an online algorithm that packs any stream of squares of total area at most Σ into the unit square? This was studied, among others, by Fekete and Hoffmann [13] and Brubach [8]. Lassak [16] and Januszewski and Lassak [15] studied higher-dimensional versions of this problem.

Milenkovich [20, 21] and Milenkovich and Daniels [22] studied generalized offline versions of the minimum area problem where the pieces are simple or convex polygons. Some algorithms have been described for computing the packing of two or three convex polygons that minimizes the perimeter or area of the convex hull or the bounding box [1, 5, 17, 23].

Alt [2] and Alt, de Berg, and Knauer [4] gave constant factor approximation algorithms for the offline versions of AreaTranslation and AreaRotation when the pieces are axis parallel rectangles or convex polygons, with translations only or arbitrary rotations allowed.

Lubachevsky and Graham [18] used computational experiments to find the rectangles of minimum area into which a given number $n \leq 5000$ of congruent circles can be packed; see also the follow-up work by Specht [24]. In another paper, Lubachevsky and Graham [19] studied the problem of minimizing the perimeter instead of the area.

Another fundamental packing problem is to find the smallest square containing a given number of *unit* squares, with arbitrary rotations allowed. A long line of mathematical research has been devoted to this problem, initiated by Erdős and Graham [12] in 1975, and it is still an active research area [10].

2 The perimeter versions

In Section 2.1, we present two online algorithms to minimize the perimeter of the bounding box: the algorithm BrickTranslation solves the problem PerimeterTranslation, where we can only translate pieces; the algorithm BrickRotation solves the problem PerimeterRotation, where also rotations are allowed. Both algorithms achieve a competitive ratio of 4. In Section 2.2, we show a lower bound of 4/3 for the version with translations and 5/4 for the version with rotations.

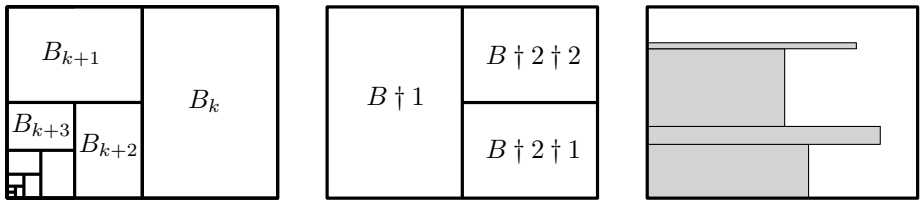


Figure 1 Left: Fundamental bricks. Middle: Splitting a brick. Right: Pieces packed in a brick.

2.1 Algorithms to minimize perimeter

Algorithm for translations

We pack the pieces into non-overlapping *bricks*; a technique first described by Januszewski and Lassak [15] which was also used by Fekete and Hoffmann [13] for the problem SQUARE-INSQUAREAREA. Let a *k-brick* be a rectangle of size $\sqrt{2}^{-k} \times \sqrt{2}^{-k-1}$ if k is even and $\sqrt{2}^{-k-1} \times \sqrt{2}^{-k}$ if k is odd. A *brick* is a k-brick for some integer k.

We tile the positive quadrant using one k-brick B_k for each integer k as in Figure 1 (left): if k is even, B_k is the k-brick with lower left corner $(0, \sqrt{2}^{-k-1})$ and otherwise, B_k is the k-brick with lower left corner $(\sqrt{2}^{-k-1}, 0)$. The bricks B_k are called the *fundamental* bricks. We define $B_{>k} := \bigcup_{i>k} B_i$ and $B_{\geq k} := B_{>k-1}$, so that $B_{>k}$ is the k-brick immediately below (if k is even) or to the left (if k is odd) of B_k.

An important property of a k-brick B is that it can be split into two $(k+1)$-bricks: $B \dagger 1$ and $B \dagger 2$; see Figure 1 (middle). We introduce a uniform naming and define $B \dagger 1$ to be the left half of B if k is even and the lower half of B if k is odd.

We define a *derived* brick recursively as follows: a derived brick is either (i) a fundamental brick B_k or (ii) $B \dagger 1$ or $B \dagger 2$, where B is a derived brick. We introduce an ordering $\prec$ of the derived k-bricks as follows. Consider two derived k-bricks D_1 and D_2 such that $D_1 \subset B_i$ and $D_2 \subset B_j$. If $i > j$, then $D_1 \prec D_2$. Else, if $i = j$ then the bricks D_1 and D_2 are both obtained by splitting the fundamental brick B_i, and the number of splits is $\ell := i - k$. Hence the bricks have the forms $D_1 = B_i \dagger b_{11} \dagger b_{12} \dagger \ldots \dagger b_{1\ell}$ and $D_2 = B_i \dagger b_{21}\, b_{22} \ldots b_{2\ell}$, where $b_{ij} \in \{1, 2\}$ for $i \in \{1, 2\}$ and $j \in \{1, \ldots, \ell\}$. We then define $D_1 \prec D_2$ if $(b_{11}, b_{12}, \ldots, b_{1\ell})$ precedes $(b_{21}, b_{22}, \ldots, b_{2\ell})$ in the lexicographic ordering.

We say that a k-brick is *suitable* for a piece p of size $w \times h$ if the width and height of the brick are at least w and h, respectively, and if that is not the case for a $(k+1)$-brick. We will always pack a given piece p in a derived k-brick that is suitable for p.

We now explain how we pack pieces into one specific brick; see Figure 1 (right). The first piece p that is packed in a brick B is placed with the lower left corner of p at the lower left corner of B. Suppose now that some other pieces $p_1, \ldots, p_i$ have been packed in B. If k is even, then $p_1, \ldots, p_i$ form a stack with the left edges contained in the left edge of B, and we place p on top of p_i (again, with the left edge of p contained in the left edge of B). Otherwise, $p_1, \ldots, p_i$ form a stack with the bottom edges contained in the bottom edge of B, and we place p to the right of p_i (again, with the bottom edge of p contained in the bottom edge of B). We say that a brick *has room* for a piece p if the packing scheme above places p within B, and it is apparent that an empty suitable brick for p has room for p.

The algorithm BRICKTRANSLATION maintains the collection $\mathcal{D}$ of non-overlapping derived bricks, such that one or more pieces have been placed in each brick in $\mathcal{D}$; see Figure 2. Before the first piece arrives, we set $\mathcal{D} := \emptyset$. Suppose that some stream of pieces have been packed, and that a new piece p appears. Choose k such that a k-brick is suitable for p. If there exists

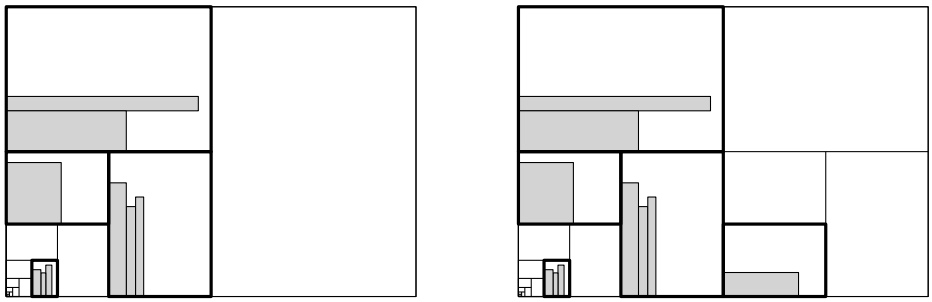


Figure 2 Left: Some pieces have been packed by the algorithm. The bricks in $\mathcal{D}$ are drawn with fat edges. Right: A new piece arrives. There is already a brick of the suitable size in $\mathcal{D}$, but there is not enough room, so a new brick of the same size is added to $\mathcal{D}$ where the piece is placed.

a derived k-brick $D \in \mathcal{D}$ such that D has room for p, then we pack p in D. Else, let D be the minimum derived k-brick (with respect to the ordering $\prec$ described before) such that D is interior-disjoint from each brick in $\mathcal{D}$; we then add D to $\mathcal{D}$ and pack p in D.

▶ **Theorem 1.** *The algorithm* BrickTranslation *has a competitive ratio strictly less than 4 for* PerimeterTranslation.

Proof. This proof relies on a careful case analysis of which bricks are contained in $\mathcal{D}$ and is deferred to the full version. ◀

Algorithm using rotations

The algorithm BrickRotation is almost identical to BrickTranslation, but with the difference that we rotate each piece so that its height is at least its width.

▶ **Theorem 2.** *The algorithm* BrickRotation *has a competitive ratio of strictly less than 4 for* PerimeterRotation.

Proof. This proof is deferred to the full version. ◀

2.2 Lower bounds

In this section we state two lower bounds worth reporting. However, their proofs are of modest interest and are deferred to the full version.

▶ **Lemma 3.** *Any algorithm for* PerimeterTranslation *has competitive ratio* $\geq 4/3$.

▶ **Lemma 4.** *Any algorithm for* PerimeterRotation *has competitive ratio* $\geq 5/4$.

3 Area versions

3.1 General lower bounds

In this section we show that, if we allow pieces to be arbitrary rectangles, we cannot bound the competitive ratio for neither AreaTranslation nor AreaRotation as a function of the area Opt of the optimal packing. However we will be able to bound the competitive ratio as a function of the total number n of pieces in the stream.

▶ **Lemma 5.** *Consider any algorithm A solving* AreaTranslation *or* AreaRotation *and let any $m \in \mathbb{N}$ and $p \in \mathbb{R}$ be given. There exists a stream of $n = m^2 + 1$ rectangles such that (i) the rectangles can be packed into a bounding box of area $2p^2$, and (ii) algorithm A produces a packing with a bounding box of area at least mp^2.*

Proof. We first feed A with m^2 rectangles of size $p \times \frac{p}{m^2}$. These rectangles have total area p^2. Let $a \times b$ be the size of the bounding box of the produced packing.

Suppose first that $a \geq \frac{p}{m}$ and $b \geq \frac{p}{m}$ hold. We then feed A with a long rectangle of size $pm^2 \times \frac{p}{m^2}$. The produced packing has a bounding box of area at least $\frac{p}{m} \cdot pm^2 = mp^2$. The optimal packing is to pack the m^2 small rectangles along the long rectangle, which would produce a packing with bounding box of size $pm^2 \times \frac{2p}{m^2} = 2p^2$.

Otherwise, we must have $b > pm$ or $a > pm$, since $ab \geq p^2$. We then feed A with a square of size $p \times p$. The produced packing has a bounding box of area at least $p \cdot pm = mp^2$. The optimal packing is obtained stacking the m^2 thin rectangles on top of the big square, which produces a packing with bounding box of size $p \times 2p = 2p^2$. ◀

▶ **Corollary 6.** *Let A be an algorithm for* AreaTranslation *or* AreaRotation*. Then A does not have a competitive ratio which is a function of* Opt*.*

Proof. Let f be any function of Opt. For any value $\text{Opt} = c$, we choose $p := \sqrt{c/2}$. We now choose $m > 2f(c)$ and obtain that the competitive ratio is at least $\frac{mp^2}{2p^2} = m/2 > f(c) = f(\text{Opt})$. ◀

▶ **Corollary 7.** *Let A be an algorithm for* AreaTranslation *or* AreaRotation*. If A has a competitive ratio of $f(n)$, where $n = |L|$ is the number of pieces in the stream, then $f(n) = \Omega(\sqrt{n})$. This holds even when all edges of the pieces are required to have length at least 1.*

Proof. We choose $p := m^2$. Then all edges have length at least 1, and the competitive ratio is at least $\frac{mp^2}{2p^2} = m/2 = \Omega(\sqrt{n})$. Here, Opt can be arbitrarily big by choosing m big enough, so it is a lower bound on the competitive ratio. ◀

3.2 Algorithms for arbitrary pieces

In this section we provide algorithms that solve AreaTranslation and AreaRotation with a competitive ratio of $O(\sqrt{n})$, where n is the total number of pieces. Thus we match the bounds provided in the previous section.

We first describe the algorithm DynBoxTrans that solves AreaTranslation. We assume to receive a stream of pieces $p_1, \ldots, p_n$ of unknown length n, such that piece p_i has size $w_i \times h_i$. For each $k \in \mathbb{Z}$, we define a rectangular box B_k with a size varying dynamically. After pieces $p_1, \ldots, p_j$ have been processed B_k has size $2^k \times T_j$, where $T_j := H_j\sqrt{j} + 7H_j$ and $H_j := \max_{i=1,\ldots,j} h_i$. We place the boxes with their bottom edges on the x-axis and in order such that the right edge of B_{k-1} is contained in the left edge of B_k; see Figure 3. Furthermore, we place the lower left corner of box B_0 at the point $(1, 0)$. It then holds that all the boxes are to the right of the point $(0, 0)$.

We say that the box B_k is *wide enough* for a piece $p_i = w_i \times h_i$ if $w_i \leq 2^k$. If a box B_k is wide enough for p_i, we can pack p_i in B_k using the online strip packing algorithm NFS_k that packs rectangles into a strip of width 2^k. The algorithm NFS_k is the *next-fit shelf algorithm* first described by Baker and Schwartz [6]. The algorithm packs pieces in *shelves* (rows), and each shelf is given a fixed height of 2^j for some $j \in \mathbb{Z}$ when it is created; see Figure 4. The width of each shelf is 2^k, since this is the width of the box B_k.

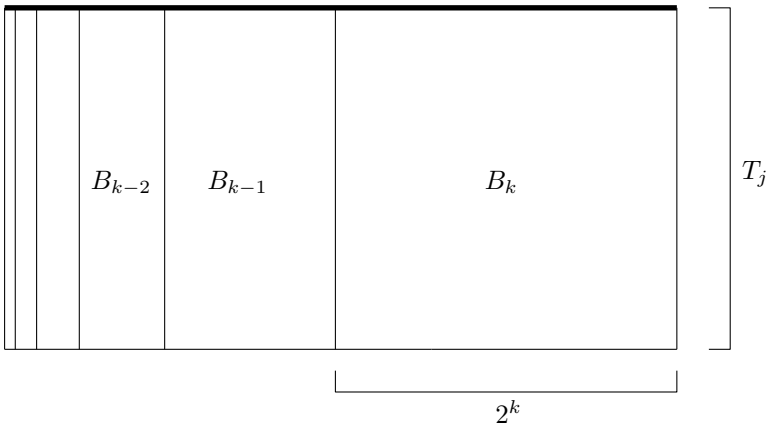


Figure 3 The algorithm DynBoxTrans packs pieces into the boxes B_k that form a row. Every box has height T_j that depends on $p_1 \dots p_j$ and is dynamically updated.

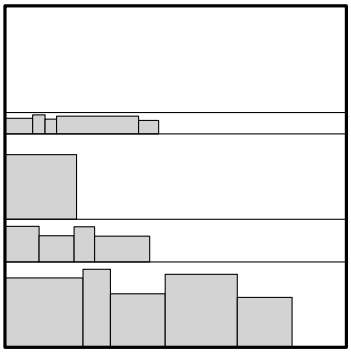


Figure 4 A packing produced by the next-fit shelf algorithm using four shelves.

A piece of height h, where $2^{j-1} < h \leq 2^j$, is packed in a shelf of height 2^j. We divide the shelves into two types. If the total width of pieces in a shelf is more than 2^{k-1} we call that shelf *dense*, otherwise we say it is *sparse*. The algorithm NFS_k places each piece as far left as possible into the currently sparse shelf of the proper height. If there is no sparse shelf of this height or the sparse shelf has not room for the piece, a new shelf of the appropriate height is created on top of the top shelf, and the piece is placed there at the left end of this new shelf. This ensures that at any point in time there exists at most one sparse shelf for each height 2^j.

If we allow the height of the box B_k to grow large enough with respect to shelves' heights, the space wasted by sparse shelves becomes negligible and we obtain a constant density strip packing, as stated in the following lemma.

▶ **Lemma 8.** *Let $\widetilde{H}$ be the total height of shelves in B_k, and H_{max} be the maximum height among pieces in B_k. If $\widetilde{H} \geq 6H_{max}$, then the pieces in B_k are packed with density at least $1/12$.*

Proof. Let $2^{m-1} < H_{max} \leq 2^m$, so that $\widetilde{H} \geq 3 \cdot 2^m$. For each $i \leq m$ we have at most one sparse shelf of height 2^i and each shelf of B_k has height at most 2^m, hence the total height of sparse shelves is at most $\sum_{i \leq m} 2^i = 2^{m+1}$, so the total height of dense shelves is at least $\widetilde{H} - 2^{m+1} \geq \widetilde{H}/3$. Thus, the total area of the dense shelves is at least $2^k \cdot \widetilde{H}/3$.

Consider a dense shelf of height 2^i. Into that shelf, we have packed pieces of height at least 2^{i-1}, and the total width of these pieces is at least 2^{k-1}. Hence, the density of pieces in the shelf is at least $1/4$. Therefore, the total area of pieces in B_k is at least $2^k \cdot \widetilde{H}/12$. On the other hand, the area of the bounding box is $2^k \cdot \widetilde{H}$, that yields the desired density. ◀

Now we are ready to describe how the algorithm works. When the first piece p_1 arrives, let $2^{k-1} < w_1 \leq 2^k$, then we pack it in the box B_k according to NFS_k and define B_k to be the *active box*. Suppose now that B_i is the active box when the piece p_j arrives, first we update the value of the threshold T_{j-1} to T_j, then we have two cases. If $w_j > 2^i$ we choose ℓ such that $2^{\ell-1} < w_j \leq 2^\ell$, pack p_j in B_ℓ and define B_ℓ to be the active box. Else, B_i is wide enough for p_j and we try to pack p_j into B_i. Since B_i has size $2^i \times T_j$ it may happen that NFS_i exceeds the threshold T_j while packing p_j, generating an overflow. In this case, instead of packing p_j in B_i, we pack p_j into B_{i+1} and define that to be the active box.

▶ **Theorem 9.** *The algorithm* DynBoxTrans *has a competitive ratio of* $O(\sqrt{n})$ *for the problem* AreaTranslation *on a stream of* n *pieces.*

Proof. First, define Σ_j as the total area of the first j pieces, $W := \max_{i=1,\dots,n} w_i$ and recall that $H_j = \max_{i=1,\dots,j} h_i$ and $T_j = H_j\sqrt{n} + 7H_j$. Let B_k be the last active box, so that we can enclose all the pieces in a bounding box of size $2^{k+1} \times T_n$, and bound the area returned by the algorithm as $\text{Alg} = O(2^k H_n \sqrt{n})$. On the other hand we are able to bound the optimal offline packing as $\text{Opt} = \Omega(\Sigma_n + WH_n)$.

If the active box never changed, then we have $2^k < 2W$ that implies $\text{Alg} = O(WH_n\sqrt{n}) = \text{Opt} \cdot O(\sqrt{n})$. Otherwise, let B_ℓ be the last active box before B_k, and p_j be the first piece put in B_k. Here we have two cases.

Case (1) $w_j > 2^\ell$ In this case we have $2^k < 2W$ that implies $\text{Alg} = O(WH_n\sqrt{n}) = \text{Opt} \cdot O(\sqrt{n})$.

Case (2) $w_j \leq 2^\ell$ In this case we have $k = \ell + 1$. Denote with $\widetilde{H_i}$ the total height of shelves in B_i. Then we have $\widetilde{H_\ell} \geq T_j - H_j = H_j\sqrt{n} + 6H_j$, otherwise we could pack p_j in B_ℓ. Thus, we can apply Lemma 8 and conclude that the box B_ℓ of size $2^\ell \times T_j$ is filled with constant density. Here we have two cases.

Case (2.1) $\widetilde{H_k} \leq T_j$ In this case we have $\text{Alg} = O(2^k T_j)$ and, thanks to the constant density packing of B_ℓ we have $\Sigma_j = \Theta(2^\ell \widetilde{H_\ell}) = \Theta(2^k T_j)$. Since $\text{Opt} \geq \Sigma_j$, we get $\text{Alg} = O(\text{Opt})$.

Case (2.2) $\widetilde{H_k} > T_j$ In this case we have $\text{Alg} = O(2^k \widetilde{H_k})$. Moreover, $\widetilde{H_k} = O(H_n + \Sigma_n/2^k)$, in fact if $2^{s-1} < H_n \leq 2^s$, then the total height of sparse shelves is $\sum_{i \leq s} 2^i = 2^{s+1} = O(H_n)$. Furthermore, dense shelves are filled with constant density, therefore their total height is at most $O(\Sigma_n/2^k)$. Finally, we need to show that $2^k = O(W\sqrt{n})$. Thanks to the constant density packing of B_ℓ, we have $2^k H_j \sqrt{j} = O(2^\ell T_j) = O(\Sigma_j)$. We can upper bound the size of every piece p_i for $i \leq j$ with $W \times H_j$ and obtain $\Sigma_j \leq n \cdot WH_j$. Plugging it in the previous estimate and dividing both sides by $H_j\sqrt{n}$ we get $2^k = O(W\sqrt{n})$. Now we have $\text{Alg} = O(2^k \widetilde{H_k}) = O(2^k H_n + \Sigma_n) = O(WH_n\sqrt{n} + \Sigma_n) = \text{Opt} \cdot O(\sqrt{n})$. ◀

The algorithm DynBoxRot is obtained from DynBoxTrans with a slight modification: before processing any piece p_i we rotate it so that $w_i \leq h_i$. In this way, it still holds that $\text{Opt} = \Omega(\Sigma_n + WH_n)$ and the proof of Theorem 9 works also for the following.

▶ **Theorem 10.** *The algorithm* DynBoxRot *has a competitive ratio of* $O(\sqrt{n})$ *for the problem* AreaRotation *on a stream of* n *pieces.*

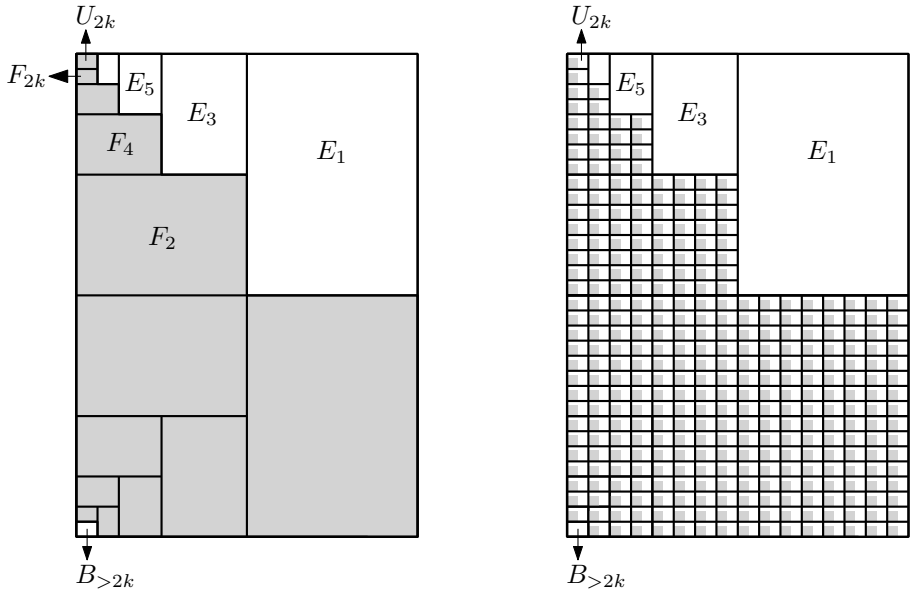


Figure 5 Left: A $2k$-packing. The grey bricks are non-empty and may have been split into smaller bricks. Right: The $2k$-packing produced by BrickTranslation when providing the algorithm with enough copies of the square S_k (the small grey squares), showing that the competitive ratio can be arbitrarily close to 6.

3.3 Bounded aspect ratio

In this section, we will consider the special case where the aspect ratio of all pieces is $\alpha = 1$, i.e., all the pieces are squares. Furthermore, we will measure the size of the packing as the area of the minimum axis-parallel bounding *square*, and we call the resulting problem SquareInSquareArea. Since we get a constant competitive ratio in this case, it follows that for other values of α and when allowing the bounding box to be a general rectangle, one can likewise achieve a constant competitive ratio. Here we give a lower bound, that is proven in the full version.

▶ **Lemma 11.** *Consider any algorithm A for the problem SquareInSquareArea. Then the competitive ratio of A is at least* $16/9$.

We are now going to analyze the competitive ratio of the algorithm BrickTranslation when it is fed with squares only. Note that a brick can never contain more than one piece. The algorithm is almost the same as the one described by Fekete and Hoffmann [13]. Their analysis proved that the algorithm is 8-competitive, with a more careful analysis we prove the following.

▶ **Theorem 12.** *The algorithm BrickTranslation has a competitive ratio of* 6 *for SquareInSquareArea. The analysis is tight.*

Proof. Suppose a stream of squares have been packed by BrickTranslation, and let Alg be the area of the bounding square of the resulting packing. Let B_k be the largest elementary brick in which a square has been placed. Suppose without loss of generality that $k = 0$, so that B_k has size $1 \times 1/\sqrt{2}$ and $B_{\geq k}$, which contains all the packed squares, has size $1 \times \sqrt{2}$.

We now recursively define a type of packing that we call a $2k$-packing, for a non-negative integer k; see Figure 5 (left). As k increases, so do the requirements to a $2k$-packing, in the sense that a $(2k+2)$-packing is also a $2k$-packing, but the other way is in general not the

case. Define $F_0 := B_{\geq 1}$ and $U_0 := B_0$. A packing is a 0-packing if pieces have been placed in U_0 (the brick U_0 may or may not have been split in smaller bricks). Hence, the considered packing is a 0-packing by the assumption that a piece has been placed in B_0. Suppose that we have defined a $2k$-packing for some integer k. A $(2k+2)$-packing is a $2k$-packing with the additional requirements that

- the brick U_{2k} has been split into $L := U_{2k} \dagger 1$ and $E_{2k+1} := U_{2k} \dagger 2$,
- the right brick E_{2k+1} is empty,
- the left brick L has been split into $F_{2k+2} := L \dagger 1$ and $U_{2k+2} := L \dagger 2$, and
- U_{2k+2} is non-empty, and thus also F_{2k+2} is non-empty.

The symbols U_j, E_j, F_j have been chosen such that the brick is a j-brick, i.e., the index tells the size of the brick.

Consider a $2k$-packing. It follows from the definition that along the top edge of $B_{\geq 0}$ from the right corner $(1, \sqrt{2})$ to the left corner $(0, \sqrt{2})$, we meet a sequence $E_1, E_3, \ldots, E_{2k-1}$ of empty bricks of decreasing size, and finally meet a non-empty brick U_{2k} which may have been split into smaller bricks.

In the full version the following claim is proven.

▷ Claim 13. If the packing is a $2k$-packing and not a $(2k+2)$-packing, then $\textsc{Alg}/\textsc{Opt} < 6$.

Since we pack a finite number of squares, the produced packing is a $2k$-packing but not a $(2k+2)$-packing for some sufficiently large k, so Claim 13 implies Theorem 12.

Moreover we prove in the full version that this analysis is tight. The idea is to show that for any given k and a small $\varepsilon > 0$, we can force the algorithm to produce a $2k$-packing, such that as $k \longrightarrow \infty$ and $\varepsilon \longrightarrow 0$, the ratio $\frac{\textsc{Alg}}{\textsc{Opt}}$ tends to 6. We use a stream where all pieces are a square S_k of size slightly more than $\sqrt{2}^{-k}/2 \times \sqrt{2}^{-k}/2$; see Figure 5 (right). ◀

3.4 More lower bounds when edges are long

We already saw in Corollary 7 that as a function of n, the competitive ratio of an algorithm for AreaTranslation or AreaRotation must be at least $\Omega(\sqrt{n})$, even when all edges have length 1. In this section, we give lower bounds in terms of Opt for the same case. Note that the assumption that the edges are long is needed for these bounds to be matched by actual algorithms, since Corollary 6 states that without the assumption, the competitive ratio cannot be bounded as a function of Opt.

▶ **Theorem 14.** *Consider any algorithm A for the problem AreaTranslation with the restriction that all edges of the given rectangles have length at least 1. If A has a competitive ratio $f(\textsc{Opt})$ as a function of Opt, then $f(\textsc{Opt}) = \Omega(\sqrt{\textsc{Opt}})$.*

▶ Remark 15. Note that when the edges are long, $\sqrt{\textsc{Opt}} = \Omega(\sqrt{n})$, so this bound is stronger than the $\Omega(\sqrt{n})$ bound of Corollary 7.

Proof of Theorem 14. For any $n \in \mathbb{N}$, we do as follows. We first provide A with n^2 unit squares. Let the bounding box of the produced packing of these squares have size $a \times b$. Assume without loss of generality that $a \leq b$, so that $b \geq n$. We now give A the rectangle $n^2 \times 1$. The optimal offline solution to this set of rectangles has a bounding box of size $n^2 \times 2$. The packing produced by A has a bounding box of size at least $n^2 \times n = \Omega(\sqrt{\textsc{Opt}}) \cdot \textsc{Opt}$. ◀

▶ **Theorem 16.** *Consider any algorithm A for the problem AreaRotation with the restriction that all edges of the given rectangles have length at least 1. If A has a competitive ratio $f(\textsc{Opt})$ as a function of Opt, then $f(\textsc{Opt}) = \Omega(\sqrt[4]{\textsc{Opt}})$.*

Proof. For any $n \in \mathbb{N}$, we do as follows. We first provide A with n^2 unit squares. Let the bounding box of the produced packing of these squares have size $a \times b$. Assume without loss of generality that $a \leq b$. If $a \geq n^{1/2}$, we give A the rectangle $1 \times n^2$. Otherwise, we have $b > n^{3/2}$, and then we give A the square $n \times n$. In either case, there is an optimal offline solution of area $2n^2$, but the bounding box of the packing produced by A has area at least $n^{5/2} = \Omega(\sqrt[4]{\textsc{Opt}}) \cdot \textsc{Opt}$. ◀

3.5 Algorithms when edges are long

In this section, we describe algorithms that match lower bounds of Section 3.4. We analyze these algorithms under the assumption that we feed them with rectangles with edges of length at least 1 (of course, any other positive constant will also work), but we require no bound on the aspect ratio. Under this assumption, we observe that DynBoxTrans has competitive ratio $O(\sqrt{\textsc{Opt}})$ for AreaTranslation. We then describe the algorithm DynBoxRot$_{\sqrt[4]{\textsc{Opt}}}$, which we prove to have competitive ratio $O(\sqrt[4]{\textsc{Opt}})$ for AreaRotation. By Theorems 14 and 16, both algorithms are optimal to within a constant factor.

In previous sections we proved lower bounds of $\Omega(\sqrt{n})$ and $\Omega(\sqrt[4]{\textsc{Opt}})$ for AreaRotation. They can be summarized stating that AreaRotation has a competitive ratio of $\Omega(\max\{\sqrt{n}, \sqrt[4]{\textsc{Opt}}\})$. The last theorem of this section, describes the algorithm DynBoxRot$_{\sqrt{n} \wedge \sqrt[4]{\textsc{Opt}}}$ that simultaneously matches both lower bounds achieving a competitive ratio of $O(\min\{\sqrt{n}, \sqrt[4]{\textsc{Opt}}\})$. At a first sight it may seem that this algorithm contradicts the lower bound of $\Omega(\max\{\sqrt{n}, \sqrt[4]{\textsc{Opt}}\})$; however this simply proves that the *edge cases* that achieve a competitive ratio of at least $\Omega(\sqrt[4]{\textsc{Opt}})$ must satisfy $\textsc{Opt} = O(n^2)$. Likewise, those for which the competitive ratio is at least $\Omega(\sqrt{n})$ satisfy $n = O(\sqrt{\textsc{Opt}})$.

Translations only

Under the long edge assumption, we have $n \leq \textsc{Opt}$. Therefore, DynBoxTrans achieves a competitive ratio of $O(\sqrt{n}) = O(\sqrt{\textsc{Opt}})$ for AreaTranslation and matches the bound stated in Theorem 14.

Rotations allowed

Now we tackle the AreaRotation problem and describe the algorithm DynBoxRot$_{\sqrt[4]{\textsc{Opt}}}$. We define the threshold function $T_j = \Sigma_j^{3/4} + 7H_j$, where $H_j = \max_{i=1,\dots,j} h_i$ and Σ_j is the total area of pieces $p_1, \dots, p_j$. DynBoxRot$_{\sqrt[4]{\textsc{Opt}}}$ is obtained by running DynBoxRot, as described in Section 3.2, employing this new threshold T_j.

▶ **Theorem 17.** *The algorithm DynBoxRot$_{\sqrt[4]{\textsc{Opt}}}$ has a competitive ratio of $O(\sqrt[4]{\textsc{Opt}})$ for the problem AreaRotation, where Opt is the area of the optimal offline packing.*

Proof. This proof is similar to the one of Theorem 9. Define $W := \max_{i=1,\dots,n} w_i$. Recall that in DynBoxRot we preprocess every piece p rotating it so that $w_p \leq h_p$, hence $W \leq \sqrt{\Sigma_n}$. Let B_k be the last active box, so that we can enclose all the pieces in a bounding box of size $2^{k+1} \times T_n$, and bound the area returned by the algorithm as $\textsc{Alg} = O(2^k H_n + 2^k \Sigma_n^{3/4})$. On the other hand we are able to bound the optimal offline packing as $\textsc{Opt} = \Omega(\Sigma_n + W H_n)$.

If the active box never changed, then we have $2^k < 2W$ that implies $\textsc{Alg} = O(W H_n + \Sigma_n^{5/4}) = \textsc{Opt} \cdot O(\sqrt[4]{\textsc{Opt}})$. Otherwise, let B_ℓ be the last active box before B_k, and p_j be the first piece put in B_k. Here we have two cases.

Case (1) $w_j > 2^\ell$ In this case we have $2^k < 2W$ that implies $\textsc{Alg} = O(WH_n + \Sigma_n^{5/4}) = \textsc{Opt} \cdot O(\sqrt[4]{\textsc{Opt}})$.

Case (2) $w_j \le 2^\ell$ In this case we have $k = \ell + 1$. Denote with $\widetilde{H_i}$ the total height of shelves in B_i. Then we have $\widetilde{H_\ell} \ge T_j - H_j = \Sigma_j^{3/4} + 6H_j$, otherwise we could pack p_j in B_ℓ. Thus, we can apply Lemma 8 and conclude that the box B_ℓ of size $2^\ell \times T_j$ is filled with constant density. Here we have two cases.

Case (2.1) $\widetilde{H_k} \le T_j$ In this case we have $\textsc{Alg} = O(2^k T_j)$ and, thanks to the constant density packing of B_ℓ we have $\Sigma_j = \Theta(2^\ell \widetilde{H_\ell}) = \Theta(2^k T_j)$. Since $\textsc{Opt} \ge \Sigma_j$, we get $\textsc{Alg} = O(\textsc{Opt})$.

Case (2.2) $\widetilde{H_k} > T_j$ In this case we have $\textsc{Alg} = O(2^k \widetilde{H_k})$. Moreover, $\widetilde{H_k} = O(H_n + \Sigma_n/2^k)$, in fact if $2^{s-1} < H_n \le 2^s$, then the total height of sparse shelves is $\sum_{i \le s} 2^i = 2^{s+1} = O(H_n)$. Furthermore, dense shelves are filled with constant density, therefore their total height is at most $O(\Sigma_n/2^k)$. Finally, we need to show that $2^k = O(\sqrt[4]{\Sigma_n})$. Thanks to the constant density packing of B_ℓ, we have $2^k \Sigma_j^{3/4} = O(2^\ell T_j) = O(\Sigma_j)$. Dividing both sides by $\Sigma_j^{3/4}$ we get $2^k = O(\Sigma_j^{1/4})$. In the end notice that, thanks to the long edge hypotheses $H_n \le \Sigma_n$ and we have $\textsc{Alg} = O(2^k \widetilde{H_k}) = O(2^k H_n + \Sigma_n) = O(\Sigma_n^{5/4}) = \textsc{Opt} \cdot O(\sqrt[4]{\textsc{Opt}})$. ◀

So far we managed to match the competitive ratio lower bounds of $\Omega(\sqrt{n})$ and $\Omega(\sqrt[4]{\textsc{Opt}})$ employing two different algorithms: DynBoxRot and DynBoxRot$_{\sqrt[4]{\textsc{Opt}}}$. A natural question is whether is it possible to match the performance of these algorithms simultaneously, having an algorithm that achieves a competitive ratio of $O(\min\{\sqrt{n}, \sqrt[4]{\textsc{Opt}}\})$. We give an affirmative answer by describing the algorithm DynBoxRot$_{\sqrt{n} \wedge \sqrt[4]{\textsc{Opt}}}$.

Again, we employ the same scheme of DynBoxRot with a different threshold function. This time the definition of T_j is slightly more involved:

$$T_j = \begin{cases} 0 & \text{if } j = 0 \\ \max\left\{T_{j-1}, \widetilde{T_j}\right\} & \text{if } j \ge 1 \end{cases} \quad \text{where} \quad \widetilde{T_j} = \begin{cases} \Sigma_j^{3/4} + 7H_j & \text{if } \Sigma_j < j^2 \\ H_j\sqrt{n} + 7H_j & \text{otherwise.} \end{cases}$$

▶ **Theorem 18.** *The algorithm* DynBoxRot$_{\sqrt{n} \wedge \sqrt[4]{\textsc{Opt}}}$ *achieves a competitive ratio of* $O(\min\{\sqrt{n}, \sqrt[4]{\textsc{Opt}}\})$ *on the problem* AreaRotation*, where* Opt *is the area of the optimal offline packing and* n *is the total number of pieces in the stream.*

Proof. This proof is similar to the one of Theorem 17 and is deferred to the full version. ◀

4 Further questions

It is natural to consider problems where the given pieces are more general, such as convex polygons. Here, we may allow the pieces to be rotated by arbitrary angles. In that case, it follows from the technique described by Alt [2] that one can obtain a constant competitive ratio for computing a packing with a minimum perimeter bounding box: For each new piece, we rotate the piece so that a diameter of the piece is horizontal. We then use the algorithm BrickRotation to pack the bounding boxes of the pieces. Since the area of each piece is at least half of the area of its bounding box, the density of the produced packing is at least half of the density of the packing of the bounding boxes. This results in an increase of the competitive ratio by a factor of at most $\sqrt{2}$.

For the problem of minimizing the perimeter of the bounding box (or convex hull) with convex polygons as pieces and only translations allowed, we do not know if it is possible to get a competitive ratio of $O(1)$, and this seems to be a very interesting question for future

research. In order to design such an algorithm, it would be sufficient to show that for some constants $\delta > 0$ and $\Sigma > 0$, there is an online algorithm that packs any stream of convex polygons of diameter at most δ and total area at most Σ into the unit square, which is in itself an interesting problem. The three-dimensional version of this question has a negative answer, even for offline algorithms: Alt, Cheong, Park, and Scharf [3] showed that for any $n \in \mathbb{N}$, there exists a finite number of 2D unit disks embedded in 3D that cannot all be packed by translation in a cube with edges of length n.

References

1 Hee-Kap Ahn and Otfried Cheong. Aligning two convex figures to minimize area or perimeter. *Algorithmica*, 62(1-2):464–479, 2012.

2 Helmut Alt. Computational aspects of packing problems. *Bulletin of the EATCS*, 118, 2016.

3 Helmut Alt, Otfried Cheong, Ji-won Park, and Nadja Scharf. Packing 2D disks into a 3D container. In *International Workshop on Algorithms and Computation (WALCOM 2019)*, pages 369–380, 2019.

4 Helmut Alt, Mark de Berg, and Christian Knauer. Approximating minimum-area rectangular and convex containers for packing convex polygons. In *23rd Annual European Symposium on Algorithms (ESA 2015)*, pages 25–34, 2015.

5 Helmut Alt and Ferran Hurtado. Packing convex polygons into rectangular boxes. In *18th Japanese Conference on Discrete and Computational Geometry (JCDCGG 2000)*, pages 67–80, 2000.

6 Brenda S. Baker and Jerald S. Schwarz. Shelf algorithms for two-dimensional packing problems. *SIAM Journal on Computing*, 12(3):508–525, 1983.

7 Allan Borodin and Ran El-Yaniv. *Online computation and competitive analysis*. Cambridge University Press, 2005.

8 Brian Brubach. Improved bound for online square-into-square packing. In *12th International Workshop on Approximation and Online Algorithms (WAOA 2014)*, pages 47–58, 2014. doi:10.1007/978-3-319-18263-6_5.

9 Henrik I. Christensen, Arindam Khan, Sebastian Pokutta, and Prasad Tetali. Approximation and online algorithms for multidimensional bin packing: A survey. *Computer Science Review*, 24:63–79, 2017. doi:10.1016/j.cosrev.2016.12.001.

10 Fan Chung and Ron Graham. Efficient packings of unit squares in a large square. *Discrete & Computational Geometry*, 2019.

11 János Csirik and Gerhard J. Woeginger. On-line packing and covering problems. In Amos Fiat and Gerhard J. Woeginger, editors, *Online Algorithms: The State of the Art*, pages 147–177. Springer, 1998. doi:10.1007/BFb0029568.

12 Paul Erdős and Ron Graham. On packing squares with equal squares. *Journal of Combinatorial Theory, Series A*, 19(1):119–123, 1975.

13 Sándor P. Fekete and Hella-Franziska Hoffmann. Online square-into-square packing. *Algorithmica*, 77(3):867–901, 2017. doi:10.1007/s00453-016-0114-2.

14 Amos Fiat and Gerhard J. Woeginger. Competitive analysis of algorithms. In Amos Fiat and Gerhard J. Woeginger, editors, *Online Algorithms: The State of the Art*, pages 1–12. Springer, 1998. doi:10.1007/BFb0029562.

15 Janusz Januszewski and Marek Lassak. On-line packing sequences of cubes in the unit cube. *Geometriae Dedicata*, 67(3):285–293, 1997.

16 Marek Lassak. On-line potato-sack algorithm efficient for packing into small boxes. *Periodica Mathematica Hungarica*, 34(1-2):105–110, 1997.

17 Hyun-Chan Lee and Tony C. Woo. Determining in linear time the minimum area convex hull of two polygons. *IIE Transactions*, 20(4):338–345, 1988. doi:10.1080/07408178808966189.

18 Boris D. Lubachevsky and Ronald L. Graham. Dense packings of congruent circles in rectangles with a variable aspect ratio. In Boris Aronov, Saugata Basu, János Pach, and Micha Sharir, editors, *Discrete and Computational Geometry: The Goodman-Pollack Festschrift*, pages 633–650. Springer, 2003.

19 Boris D. Lubachevsky and Ronald L. Graham. Minimum perimeter rectangles that enclose congruent non-overlapping circles. *Discrete Mathematics*, 309(8):1947–1962, 2009. doi:10.1016/j.disc.2008.03.017.

20 Victor J. Milenkovic. Translational polygon containment and minimal enclosure using linear programming based restriction. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of Computing (STOC 1996)*, pages 109–118, 1996.

21 Victor J. Milenkovic. Rotational polygon containment and minimum enclosure using only robust 2D constructions. *Computational Geometry*, 13(1):3–19, 1999. doi:10.1016/S0925-7721(99)00006-1.

22 Victor J. Milenkovic and Karen Daniels. Translational polygon containment and minimal enclosure using mathematical programming. *International Transactions in Operational Research*, 6(5):525–554, 1999. doi:10.1111/j.1475-3995.1999.tb00171.x.

23 Dongwoo Park, Sang Won Bae, Helmut Alt, and Hee-Kap Ahn. Bundling three convex polygons to minimize area or perimeter. *Computational Geometry*, 51:1–14, 2016. doi:10.1016/j.comgeo.2015.10.003.

24 E. Specht. High density packings of equal circles in rectangles with variable aspect ratio. *Computers & Operations Research*, 40(1):58–69, 2013. doi:10.1016/j.cor.2012.05.011.

25 Rob van Stee. SIGACT news online algorithms column 20: the power of harmony. *SIGACT News*, 43(2):127–136, 2012. doi:10.1145/2261417.2261440.

26 Rob van Stee. SIGACT news online algorithms column 26: Bin packing in multiple dimensions. *SIGACT News*, 46(2):105–112, 2015. doi:10.1145/2789149.2789167.

Complexity of Maximum Cut on Interval Graphs

Ranendu Adhikary
Department of Mathematics, Jadavpur University, Kolkata, India

Kaustav Bose
Department of Mathematics, Jadavpur University, Kolkata, India

Satwik Mukherjee
Department of Mathematics, Jadavpur University, Kolkata, India

Bodhayan Roy
Department of Mathematics, Indian Institute of Technology Kharagpur, Kharagpur, India

Abstract

We resolve the longstanding open problem concerning the computational complexity of Max Cut on interval graphs by showing that it is NP-complete.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness; Mathematics of computing → Combinatorial optimization

Keywords and phrases Maximum cut, Interval graph, NP-complete

Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.7

Related Version *Full Version*: https://arxiv.org/abs/2006.00061

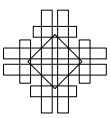
Funding *Ranendu Adhikary*: supported by Council of Scientific and Industrial Research (CSIR), Govt. of India.
Kaustav Bose: supported by National Board for Higher Mathematics (NBHM), Department of Atomic Energy (DAE), Govt. of India.
Satwik Mukherjee: supported by Council of Scientific and Industrial Research (CSIR), Govt. of India.
Bodhayan Roy: supported by an ISIRD Grant from Sponsored Research and Industrial Consultancy, IIT Kharagpur.

Acknowledgements We would like to thank the anonymous reviewers for their comments and suggestions which helped us to improve the paper.

1 Introduction

For a graph $G = (V, E)$, a *cut* is a partition of V into two disjoint subsets. Any cut determines a *cut set* which is the set of all edges that have one endpoint in one partition and the other endpoint in the other partition. The *size* of a cut is the cardinality of its cut set. The maximum cut problem or Max Cut asks for a cut of maximum size. Max Cut is a fundamental and well-known NP-complete problem [17]. The weighted version of the problem is one of Karp's original 21 NP-complete problems [25]. Besides its theoretical importance, it has applications in VLSI circuit design [11], statistical physics [3] etc. Max Cut remains NP-hard even for cubic graphs [4], split graphs [7], co-bipartite graphs [7], unit disk graphs [15] and total graphs [20]. On the positive side, polynomial time algorithms are known for planar graphs [21], line graphs [20], graphs not contractible to K_5 [2] and graphs with bounded treewidth [7].

It is well known that many classical NP-complete problems like colourability [19], Hamiltonian cycle [26], minimum dominating set [12], minimum feedback vertex set [29], minimum vertex cover [30] and maximum clique [22] are polynomial time solvable for interval graphs. This is because interval graphs are well structured graphs with many nice properties and




decomposition models that are often exploited to design efficient dynamic programming or greedy algorithms. Few problems that are known to be NP-hard in interval graphs include optimal linear arrangement [14], achromatic number [5], harmonious colouring [1], geodetic set [10], minimum sum colouring [31], metric dimension [16], identifying code [16] and locating-dominating set [16]. The class of interval graphs is widely regarded as an important graph class with many real-world applications. Interval graphs arise naturally in modelling problems that involve temporal reasoning, e.g scheduling problems. Interval graphs are also extensively used in bioinformatics (e.g. DNA mapping [32], protein sequencing [24]) and mathematical biology (e.g. food webs in population biology [13]).

Surprisingly, the computational complexity of Max Cut for interval graphs is a long-standing open problem. The first time that the problem was mentioned as open was probably in 1985 [23]. No polynomial time algorithm is known even for the subclass of unit interval graphs. There are two previous works [8, 9] reporting polynomial time algorithms solving Max Cut for unit interval graphs. However, both algorithms were later reported to be incorrect [6, 28]. In this paper, we show that Max Cut is NP-complete for interval graphs.

2 Preliminaries

For any simple undirected graph $G = (V, E)$, a *cut* is a partition of V into two disjoint subsets A and B, i.e., $V = A \cup B$ and $A \cap B = \emptyset$. The corresponding *cut set* is the set of all edges that have one endpoint in A and the other endpoint in B, i.e., the set $\{(u, v) \in E \mid (u \in A, v \in B) \vee (u \in B, v \in A)\}$. The *size* of the cut is the cardinality of its cut set. A typical instance of the decision version of Max Cut consists of a simple undirected graph $G = (V, E)$ and an integer k such that $1 \leq k \leq |E|$. (G, k) is an yes-instance of Max Cut if and only if G has a cut of size at least k.

Interval graphs are the intersection graphs of intervals on the real line. Formally, $G = (V, E)$ is said to be an *interval graph* if there is a set S of intervals on the real line and a bijection $\varphi : V \longrightarrow S$ such that $u, v \in V$ are adjacent if and only if $\varphi(u) \cap \varphi(v) \neq \emptyset$.

3 NP-Completeness

In this section, we show that Max Cut is NP-complete on interval graphs. Max Cut is known to be NP-complete on cubic graphs [4]. We reduce Max Cut on cubic graphs to Max Cut on interval graphs.

3.1 Construction of the Reduction Graph

Let (G, x) be an instance of Max Cut where $G = (V, E)$ is a cubic graph. Let $|V| = n$ and hence $|E| = \frac{3}{2}n$. We shall reduce it to an equivalent instance $(G', f(x))$ of Max Cut where $G' = (V', E')$ is an interval graph. The construction of G' is outlined in the following. $G' = (V', E')$ is described as the intersection graph of a set of intervals on the real line and the vertices of G' are referred to as intervals.

1. Fix an arbitrary ordering of the vertices and edges of $G = (V, E)$ as $v_1, v_2, \ldots, v_n, e_1, e_2, \ldots, e_m$. We shall write any edge $e \in E$ as an ordered pair of vertices that respects the following convention. If e is an edge between v_i and v_j, where $i < j$, then we shall write $e = (v_i, v_j)$ (not $e = (v_j, v_i)$).
2. For each vertex $v \in V$, we construct a V-gadget $\mathcal{G}(v)$ and for each edge $e \in E$, we construct an E-gadget $\mathcal{G}(e)$. They are shown in Fig. 1. The structure of a V-gadget is identical to that of an E-gadget, the only difference is their size. Each V-gadget (E-gadget)

consists of q (resp. q') left long intervals, p (resp. p') left short intervals, q (resp. q') right long intervals and p (resp. p') right short intervals. The left long intervals and the right long intervals of a V-gadget (E-gadget) all intersect each other to form a clique of size $2q$ (resp. $2q'$). All left short intervals of a V-gadget (E-gadget) are mutually disjoint and each of them intersect only the q (resp. q') left long intervals. Similarly all right short intervals of a V-gadget (E-gadget) are mutually disjoint and each of them intersect only the q (resp. q') right long intervals. Therefore, the number of edges in each V-gadget (E-gadget) is $q(2q-1)+2pq$ (resp. $q'(2q'-1)+2p'q'$).

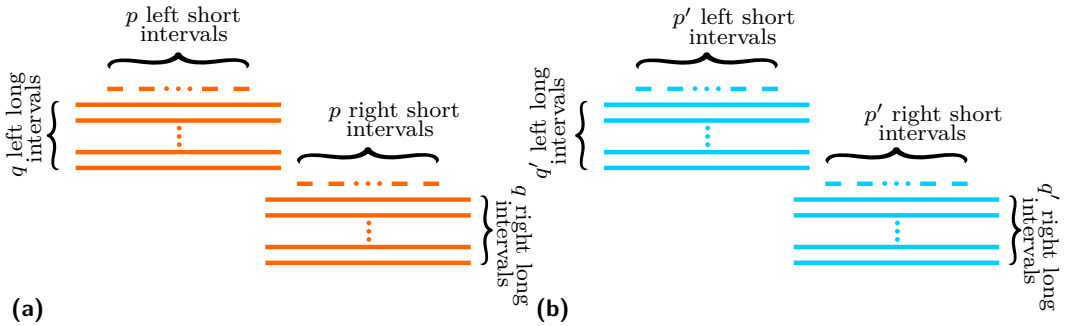


Figure 1 a) A V-gadget. b) An E-gadget.

3. We set $q = 200n^3 + 1$, $p = 2q + 7n$, $q' = 10n^2 + 1$, $p' = 2q' + 7n$, where n is the number of vertices in G. Note that the following inequalities hold:
   a. $p > 2q > 2p' > 4q' > 9n^2$
   b. $q^2 > (p-q)6n$
   c. $q'^2 > (p'-q')6n$
   d. $q > 3(p'+q')n + 9n^2$
4. There are a total of n V-gadgets, and $3n/2$ E-gadgets. All $5n/2$ gadgets are arranged in the following order as shown in Fig. 2 : $\mathcal{G}(v_1), \mathcal{G}(v_2), \ldots, \mathcal{G}(v_n), \mathcal{G}(e_1), \mathcal{G}(e_2), \ldots, \mathcal{G}(e_{3n/2})$. No two intervals belonging to different gadgets intersect.
5. To establish relationships between the V-gadgets and E-gadgets we introduce $6n$ link intervals (See Fig. 2). Link intervals connect V-gadgets to E-gadgets. This will be described in the next point. A link interval can intersect a gadget in four different ways as described in the following.
   - A link interval is said to *cover a gadget* if it intersects all intervals of the gadget. (See Fig. 3a)
   - A link interval is said to *intersect a V-gadget in the first manner* if it intersects only the q right long intervals of the V-gadget. (See Fig. 3b).
   - A link interval is said to *intersect an E-gadget in the second manner* if it intersects only the p' left long intervals of the gadget. (See Fig. 3c).
   - A link interval is said to *intersect an E-gadget in the third manner* if it intersects only the q' left long intervals and the p' left short intervals of the gadget. (See Fig. 3d).
6. For each edge $e = (v_i, v_j) \in E$, we introduce four link intervals: 1) a pair intersecting $\mathcal{G}(v_i)$ in the first manner and $\mathcal{G}(e)$ in the second manner, and 2) another pair intersecting $\mathcal{G}(v_j)$ in the first manner and $\mathcal{G}(e)$ in the third manner (See Fig. 4). Note that since G is cubic, the total number of link intervals covering a V-gadget is $6k$ for some integer k, where k may vary from 0 to $n-1$. Similarly, the total number of link intervals covering an E-gadget is $4k$ for some integer k, where k may vary from 0 to $3n/2 - 1$. Also, the total number of link intervals intersecting a V-gadget in the first manner is 6.

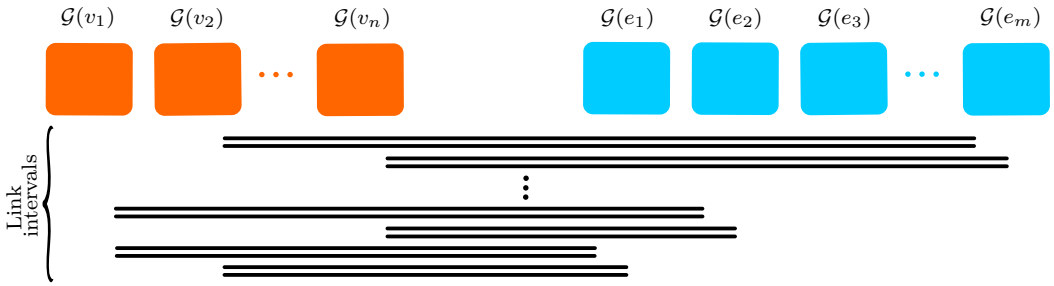
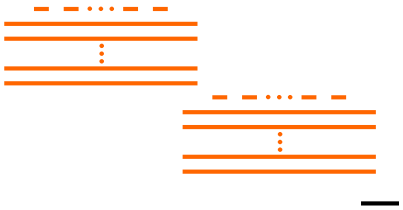
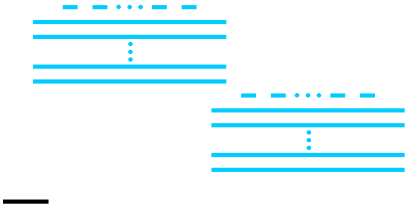


Figure 2 Arrangement of the gadgets and the link intervals.

(a) A gadget is covered by a link interval.



(b) A link interval intersects a V-gadget in the first manner.



(c) A link interval intersects an E-gadget in the second manner.

(d) A link interval intersects an E-gadget in the third manner.

Figure 3 Illustrations showing the four different ways a link interval can intersect a gadget.

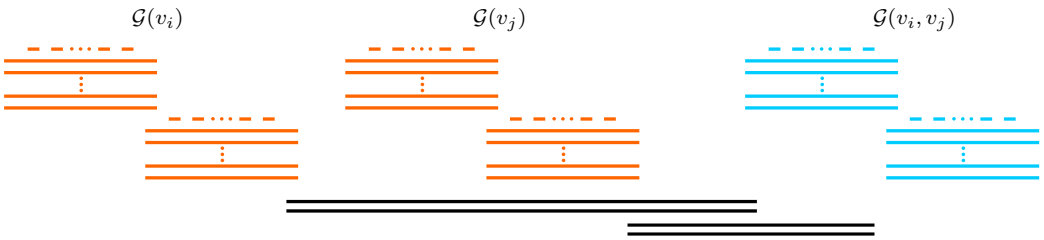


Figure 4 link intervals connecting an E-gadget $\mathcal{G}((v_i, v_j))$ with V-gadgets $\mathcal{G}(v_i)$ and $\mathcal{G}(v_j)$.

3.2 Properties of the Reduction Graph

In this section, we study some properties of the interval graph G' constructed from G in the previous section. We consider a partition of vertices of G' that yields a maximum cut. To prove that the partition satisfies some properties, in general we show that if it does not satisfy those properties, then the size of the corresponding cut can be increased, contradicting the maximality of the cut.

Now consider a maximum cut of G' with the vertices partitioned into subsets A and B. We show in the next lemma that for every vertex gadget $\mathcal{G}(v_i)$, either A or B contains all of its left short intervals. The same holds for the right short intervals of $\mathcal{G}(v_i)$.

▶ **Lemma 1.** *If a partition of G' yields a maximum cut, then for any V-gadget $\mathcal{G}(v_i)$, all of its left short intervals lie in the same subset. The same holds for its right short intervals.*

Proof. Consider a maximum cut of G' that partitions its vertices into subsets A and B. Let LL_i^A and LL_i^B denote the subset of left long intervals of $\mathcal{G}(v_i)$ in A and B respectively. Denote by OL_i^A (resp. OL_i^B) the set of all link intervals that cover $\mathcal{G}(v_i)$ and lie in subset A (resp. B). Without loss of generality, let the following direction of inequality hold:

$$| LL_i^A | + | OL_i^A | > | LL_i^B | + | OL_i^B |$$

Note that the inequality must be strict since the sum of the number of left long intervals of $\mathcal{G}(v_i)$ and the number of link intervals covering $\mathcal{G}(v_i)$ is $6k+q$ $(0 \leq k \leq n-1)$, which is odd since q is odd. Suppose that a left short interval of $\mathcal{G}(v_i)$ is in A. Recall that the left long intervals of $\mathcal{G}(v_i)$ and the link intervals covering $\mathcal{G}(v_i)$ are the only intervals that a left short interval of $\mathcal{G}(v_i)$ intersects. Then due to the above inequality, moving the left short interval to B increases the number of cut edges. This contradicts the fact that the partition yields a maximum cut. Hence, all left short intervals of $\mathcal{G}(v_i)$ must be in B. Using similar arguments we can show that all right short intervals of $\mathcal{G}(v_i)$ must be same subset. ◀

In the following lemma, we prove a property of the long intervals of each vertex gadget, akin to the property of the short intervals proved in the previous lemma.

▶ **Lemma 2.** *If a partition of G' yields a maximum cut, then for any V-gadget $\mathcal{G}(v_i)$, all its left long intervals lie in the same subset. The same holds for its right long intervals.*

Proof. Consider a maximum cut of G' that partitions its vertices into subsets A and B. By Lemma 1, all of the short intervals on the same side of $\mathcal{G}(v_i)$ belong to the same subset. Without loss of generality, we consider two cases, where (a) all the left short intervals of $\mathcal{G}(v_i)$ are in A, and all the right short intervals of $\mathcal{G}(v_i)$ are in B, and (b) all the short intervals of $\mathcal{G}(v_i)$ are in A.

First consider Case (a), where all the left short intervals of $\mathcal{G}(v_i)$ belong to A, and all the right short intervals of $\mathcal{G}(v_i)$ belong to B. Suppose that a left long interval of $\mathcal{G}(v_i)$ is in A. Then moving it to B results in losing at most $2q-1$ cut edges due to its intersections with other long intervals of $\mathcal{G}(v_i)$, and at most $6n$ cut edges due to its intersections with the link intervals of $\mathcal{G}(v_i)$. However, we gain at most p cut edges. Since $p = 2q+7n$, the quantity $p-(2q-1+6n)$ is positive, hence the size of the cut increases. This contradicts the fact that the partition yields a maximum cut. Hence, all left long intervals of $\mathcal{G}(v_i)$ must be in B.

Now consider Case (b), where all the short intervals of $\mathcal{G}(v_i)$ belong to A. It can be seen that the above argument is also applicable in this case, and the claim holds. ◀

In the following lemma, we consolidate the results obtained above into a complete partition of a vertex gadget in a maximum cut.

▶ **Lemma 3.** *If a partition of G' yields a maximum cut, then for any V-gadget $\mathcal{G}(v_i)$, all the left long and right short intervals are in one subset, while all the right long and left short intervals are in the other.*

Proof. Consider a maximum cut of G' that partitions its vertices into subsets A and B. Then without loss of generality, by Lemma 1, either (a) all left short intervals of $\mathcal{G}(v_i)$ are in A and all right short intervals of $\mathcal{G}(v_i)$ are in B, or (b) all the short intervals of $\mathcal{G}(v_i)$ are in A.

First consider Case (a), i.e. $\mathcal{G}(v_i)$ has all its left short intervals in A and right short intervals in B. Then it follows from the proof of Lemma 2 that all left long intervals of $\mathcal{G}(v_i)$ are in B and all right long intervals of $\mathcal{G}(v_i)$ must be in A, as claimed.

Now consider Case (b), i.e. $\mathcal{G}(v_i)$ has all its short intervals in A. Since all the short intervals of $\mathcal{G}(v_i)$ are in A, it implies from the proof of 2 that all the long intervals of $\mathcal{G}(v_i)$ are in B. We move all the right short intervals of $\mathcal{G}(v_i)$ to B and all right long intervals of $\mathcal{G}(v_i)$ to A. Due to their intersections with link intervals, this removes at most $(p-q)6n$ edges from the cut. But due to the intersections among the left and right long intervals, it also adds at least q^2 edges to the cut. Since by our choice of q and p, we have $q^2 - (p-q)6n > 0$, the total number of edges in the cut increases. This contradicts the fact that the partition yields a maximum cut and hence this case is impossible. ◀

It can be seen that V-gadgets and E-gadgets are structurally similar, and only their intersections with the link intervals can possibly be the cause of any different partitioning in a maximum cut. We address this point in the following lemma and show that E-gadgets too in fact admit a partition similar to that of V-gadgets.

▶ **Lemma 4.** *Lemma 3 holds for E-gadgets of G' as well.*

Proof. Consider a maximum cut of G' that partitions its vertices into subsets A and B. We modify the proof of Lemma 1 a little, so that Lemma 1 holds for E-gadgets as well. Consider an E-gadget $\mathcal{G}(e_i)$ of G'. Observe that the proof holds for the right short intervals of $\mathcal{G}(e_i)$, since any link interval that intersects the right short intervals of an E-gadget, must also cover the E-gadget. But the left short intervals of each E-gadget are intersected by two link intervals in the third manner. Then denote by $OL_i'^A$ (resp. $OL_i'^B$) the set of all link intervals that cover $\mathcal{G}(v_i)$ or intersect $\mathcal{G}(v_i)$ in the third manner, and lie in subset A (resp. B). Let LL_i^A and LL_i^B denote the subset of left long intervals of $\mathcal{G}(v_i)$ in A and B respectively, as before. Again, without loss of generality we have the following inequality.

$$| LL_i^A | + | OL_i'^A | > | LL_i^B | + | OL_i'^B |$$

The rest of the proof is similar to that of 1, and it can be seen that the claim holds. The proof of Lemma 2 for E-gadgets remains the same as for V-gadgets. Lemmas 1 and 2 along with the choice of p' and q', imply Lemma 3 for E-gadgets as well. ◀

▶ **Lemma 5.** *G has a cut of size at least x if and only if G' has a cut of size at least $(2pq+q^2)n + \frac{3}{2}(2p'q'+q'^2)n + 3(n-1)(n-2)(p+q) + 3n(\frac{3}{2}n-1)(p'+q') + 6nq + 3np' + 2xq'$.*

Proof. First suppose that G has a cut of size at least x. Denote the subsets in the partition of the vertices of G by C and D. We partition the vertices of G' as follows. If a vertex v_i of G is in C, then in the corresponding V-gadget $\mathcal{G}(v_i)$ of G', all left short intervals and right long intervals are placed in A, all right short intervals and left long intervals are placed in B. Finally, all link intervals intersecting $\mathcal{G}(v_i)$ in the first manner are placed in B. If v_i is in D instead, then all the above placements of intervals are swapped. Recall that for each

E-gadget exactly two link intervals intersect it in the second manner and exactly two link intervals intersect it in the third manner. If the link intervals that intersect an E-gadget in the third manner is in A, then we place the left short intervals and right long intervals of the E-gadget in B, and the left long intervals and right short intervals in A. If the link intervals are in B, then the placements of the intervals are swapped.

Due to the above placement of intervals in A and B, the number of cut edges obtained internally from all the V-gadgets and E-gadgets of G' are $(2pq + q^2)n$ and $\frac{3}{2}(2p'q' + q'^2)n$ respectively. The number of cut edges formed by the V-gadgets and the link intervals that cover them is $3(n-1)(n-2)(p+q)$. The number of cut edges formed by the E-gadgets and the link intervals covering them is $3n(\frac{3}{2}n - 1)(p' + q')$. For each V-gadget, the link intervals intersecting it in the first manner give $6q$ cut edges, resulting in a total of $6nq$ cut edges. Each link interval that intersects an E-gadget in the third manner gives p' cut edges, thus we have $3np'$ in total. However, a link interval that intersects an E-gadget in the second manner can produce cut edges from the E-gadget only when the other link interval mentioned above is in a different subset, i.e. the vertices of G corresponding to the V-gadgets of these link intervals are in C and D, and produce a cut edge. This means that such link intervals produce at least $2xq'$ cut edges in total, proving the forward direction of the claim.

Now we prove the backward direction of the claim. Assume that G' has a cut of size at least $(2pq+q^2)n+\frac{3}{2}(2p'q'+q'^2)n+3(n-1)(n-2)(p+q)+3n(\frac{3}{2}n-1)(p'+q')+6nq+3np'+2xq'$. So the size of a maximum cut of G' is at least this much. Consider a maximum cut of G' that partitions its intervals into two disjoint subsets A and B. By Lemma 3, for each V-gadget, all the left long and right short intervals are in one subset, while all the right long and left short intervals are in the other. Corresponding to this cut of G', we define a cut of G in the following way. If the left long and right short intervals of $\mathcal{G}(v_i)$ are in A (resp. B), then we put v_i in C (resp. D). Let y be the size of the cut $C \cup D$. We have to show that $y \geq x$.

Due to Lemma 3 and 4, the internal cut edges of V-gadgets and E-gadgets, and the cut edges formed between gadgets and the link intervals that cover them amount to $(2pq + q^2)n + \frac{3}{2}(2p'q' + q'^2)n + 3(n-1)(n-2)(p+q) + 3n(\frac{3}{2}n - 1)(p' + q')$ cut edges in total. Hence, the remaining $6nq + 3np' + 2xq'$ cut edges are obtained from the partial intersections of the link intervals with the V-gadgets and E-gadgets, and the intersections among link intervals. The number of cut edges among the link intervals is not more than $(3n)^2 = 9n^2$. The partial intersections between link intervals and V-gadgets can contribute at most $6nq$ cut edges. Note that the partial intersections between link intervals and E-gadgets, and intersections among the link intervals cannot give more than $3(p' + q')n + 9n^2$ cut edges. Since $q > 3(p' + q')n + 9n^2$, it implies that exactly $6nq$ of the remaining cut edges are obtained from link intervals intersecting V-gadgets in the first manner. This happens when for each V-gadget, the link intervals intersecting it in the first manner are all in the subset which contains the left long and right short intervals of the gadget. Hence, the placement of the intervals of the V-gadget in the subsets A and B (and hence the placement of the corresponding vertex of G in C or D) determines the placements of the link intervals.

The remaining $3np' + 2xq'$ cut edges should come from the partial intersections of the link intervals with the E-gadgets, and the intersections among link intervals. We show that this is not possible if $y < x$. For this, consider an E-gadget $\mathcal{G}(v_i, v_j)$. Let ℓ_i, ℓ'_i be the two link intervals from $\mathcal{G}(v_i)$ that intersect $\mathcal{G}(v_i, v_j)$ in the second manner and ℓ_j, ℓ'_j be the two link intervals from $\mathcal{G}(v_j)$ that intersect $\mathcal{G}(v_i, v_j)$ in the third manner. Consider the following cases: ℓ_i, ℓ'_i, ℓ_j, ℓ'_j are in the same subset (Case 1), say ℓ_i, ℓ'_i, $\ell_j, \ell'_j \in A$ and ℓ_i, ℓ'_i are in one subset and ℓ_j, ℓ'_j are in the other (Case 2), say, $\ell_i, \ell'_i \in A$, $\ell_j, \ell'_j \in B$. In Case 2, the edge (e_i, e_j) appears in the cut set of $C \cup D$, while in Case 1, it does not. For each case, we have two subcases as described in the following.

Case 1a. A contains $\ell_i, \ell'_i, \ell_j, \ell'_j$ and the left long and right short intervals of $\mathcal{G}(v_i, v_j)$. B contains the right long and left short intervals of $\mathcal{G}(v_i, v_j)$. Hence, the intersections between $\mathcal{G}(v_i, v_j)$ and $\ell_i, \ell'_i, \ell_j, \ell'_j$ give $2p'$ cut edges.

Case 1b. A contains $\ell_i, \ell'_i, \ell_j, \ell'_j$ and the right long and left short intervals of $\mathcal{G}(v_i, v_j)$. B contains the left long and right short intervals of $\mathcal{G}(v_i, v_j)$. Hence, the intersections between $\mathcal{G}(v_i, v_j)$ and $\ell_i, \ell'_i, \ell_j, \ell'_j$ give $4q'$ cut edges.

Case 2a. A contains ℓ_i, ℓ'_i and the left long and right short intervals of $\mathcal{G}(v_i, v_j)$. B contains ℓ_j, ℓ'_j and the right long and left short intervals of $\mathcal{G}(v_i, v_j)$. Hence, the intersections between $\mathcal{G}(v_i, v_j)$ and $\ell_i, \ell'_i, \ell_j, \ell'_j$ give $2q'$ cut edges.

Case 2b. A contains ℓ_i, ℓ'_i and the right long and left short intervals of $\mathcal{G}(v_i, v_j)$. B contains ℓ_j, ℓ'_j and the left long and right short intervals of $\mathcal{G}(v_i, v_j)$. Hence, the intersections between $\mathcal{G}(v_i, v_j)$ and $\ell_i, \ell'_i, \ell_j, \ell'_j$ give $2p' + 2q'$ cut edges.

Therefore, we see that an E-gadget gives at most $2p'$ cut edges from its partial intersections with link intervals if the link intervals belong to the same subset (since $2p' > 4q'$), and at most $2(p'+q')$ cut edges if the link intervals belong to different subsets (since $2p'+2q' > 2q'$). Notice that the later case occurs for exactly y E-gadgets. The number of cut edges obtained from the partial intersections of E-gadgets with link intervals is at most $2p'(\frac{3n}{2} - y) + 2(p' + q')y = 3np' + 2yq'$. Hence if $y < x$, then at least $2(x - y)q' > 2q'$ cut edges must come from the intersections among the link intervals. But this is not possible as $2q' > 9n^2$. Hence $y \geq x$ as required. ◀

▶ **Theorem 6.** *Max Cut is NP-complete on interval graphs.*

Proof. It can be checked in polynomial time if a given partition of an interval graph produces a cut of a given size. Thus the problem is in NP. The construction of G' from G clearly takes polynomial time. The NP-hardness follows from Lemma 5. ◀

4 Concluding Remarks

In this paper, we have settled the question of computational complexity of Max Cut on interval graphs. However, the question of whether Max Cut is polynomial-time solvable or NP-hard on unit interval graphs still remains open. For an NP-hardness reduction, a possible approach might be to reduce Max Cut on interval graphs to Max Cut on unit interval graphs. An interval can be transformed into a sequence of unit intervals by replacing it with a start and end interval, with "bunches" of unit intervals within (See Fig. 5). It is easy to see that for such a standalone gadget, an alternating assignment of the bunches to the two subsets yields a Max Cut. However, when multiple such gadgets of different sizes are brought together to represent the whole interval graph for the reduction, such a partition does not necessarily correspond to a partition in the original interval graph.

Figure 5 Transformation of an interval into a sequence of unit intervals in a possible reduction from Max Cut on interval graphs to Max Cut on unit interval graphs.

Another direction for future work is to find approximation algorithms for Max Cut interval graphs. In general, polynomial-time approximation algorithm for Max Cut with the best known approximation ratio is by Goemans and Williamson [18] which achieves an approximation ratio ≈ 0.878. Assuming the Unique Games conjecture [27], this is the best possible approximation ratio. An interesting question is whether this can be bettered for interval graphs or unit interval graphs. A possible approach could be the following greedy method. We first compute a unit interval representation the graph. In the first step, the leftmost interval is put in A, then the leftmost interval not intersecting that interval is put in A, and so on. In the second step, among the remaining intervals, we consider the ones that intersect the most number of intervals put in A. The leftmost such interval is put in B, then the leftmost of them not intersecting that interval is put in B, and so on. We repeat this until all intervals are placed, i.e., in each odd (resp. even) step an independent set of intervals, each of which intersect the most number of intervals put in B (resp A) thus far, are put in A (resp. B). It is not clear to us how efficient this is, but the following is the worst example that we have found so far which gives an approximation ratio of 0.9375. Consider a graph $G = (V, E)$ with $8a$ vertices $\{v_1, \ldots, v_{8a}\}$ where the first $6a$ vertices are all adjacent to each other and the last $6a$ vertices are all adjacent to each other. The greedy algorithm gives a cut of size $15a^2$, while the maximum cut is of size $16a^2$.

References

1. Katerina Asdre, Kyriaki Ioannidou, and Stavros D. Nikolopoulos. The harmonious coloring problem is NP-complete for interval and permutation graphs. *Discret. Appl. Math.*, 155(17):2377–2382, 2007. doi:10.1016/j.dam.2007.07.005.
2. Francisco Barahona. The max-cut problem on graphs not contractible to K_5. *Operations Research Letters*, 2(3):107–111, 1983. doi:10.1016/0167-6377(83)90016-0.
3. Francisco Barahona, Martin Grötschel, Michael Jünger, and Gerhard Reinelt. An application of combinatorial optimization to statistical physics and circuit layout design. *Operations Research*, 36(3):493–513, 1988.
4. Piotr Berman and Marek Karpinski. On some tighter inapproximability results. In Jirí Wiedermann, Peter van Emde Boas, and Mogens Nielsen, editors, *Automata, Languages and Programming, 26th International Colloquium, ICALP'99, Prague, Czech Republic, July 11-15, 1999, Proceedings*, volume 1644 of *Lecture Notes in Computer Science*, pages 200–209. Springer, 1999. doi:10.1007/3-540-48523-6_17.
5. Hans L. Bodlaender. Achromatic number is NP-complete for cographs and interval graphs. *Inf. Process. Lett.*, 31(3):135–138, 1989. doi:10.1016/0020-0190(89)90221-4.
6. Hans L. Bodlaender, Celina M. H. de Figueiredo, Marisa Gutierrez, Ton Kloks, and Rolf Niedermeier. Simple max-cut for split-indifference graphs and graphs with few P_4s. In Celso C. Ribeiro and Simone L. Martins, editors, *Experimental and Efficient Algorithms, Third International Workshop, WEA 2004, Angra dos Reis, Brazil, May 25-28, 2004, Proceedings*, volume 3059 of *Lecture Notes in Computer Science*, pages 87–99. Springer, 2004. doi:10.1007/978-3-540-24838-5_7.
7. Hans L Bodlaender and Klaus Jansen. On the complexity of the maximum cut problem. *Nordic Journal of Computing*, 7(1):14–31, 2000.
8. Hans L. Bodlaender, Ton Kloks, and Rolf Niedermeier. SIMPLE MAX-CUT for unit interval graphs and graphs with few P_4s. *Electron. Notes Discret. Math.*, 3:19–26, 1999. doi:10.1016/S1571-0653(05)80014-9.
9. Arman Boyaci, Tínaz Ekim, and Mordechai Shalom. A polynomial-time algorithm for the maximum cardinality cut problem in proper interval graphs. *Inf. Process. Lett.*, 121:29–33, 2017. doi:10.1016/j.ipl.2017.01.007.

10 Dibyayan Chakraborty, Sandip Das, Florent Foucaud, Harmender Gahlawat, Dimitri Lajou, and Bodhayan Roy. Algorithms and complexity for geodetic sets on planar and chordal graphs. In Yixin Cao, Siu-Wing Cheng, and Minming Li, editors, *31st International Symposium on Algorithms and Computation, ISAAC 2020, December 14-18, 2020, Hong Kong, China (Virtual Conference)*, volume 181 of *LIPIcs*, pages 7:1–7:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ISAAC.2020.7.

11 K. C. Chang and David Hung-Chang Du. Efficient algorithms for layer assignment problem. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, 6(1):67–78, 1987. doi:10.1109/TCAD.1987.1270247.

12 Maw-Shang Chang. Efficient algorithms for the domination problems on interval and circular-arc graphs. *SIAM J. Comput.*, 27(6):1671–1694, 1998. doi:10.1137/S0097539792238431.

13 Joel E Cohen and David W Stephens. *Food webs and niche space*. Princeton University Press, 1978.

14 Johanne Cohen, Fedor V. Fomin, Pinar Heggernes, Dieter Kratsch, and Gregory Kucherov. Optimal linear arrangement of interval graphs. In Rastislav Kralovic and Pawel Urzyczyn, editors, *31st International Symposium on Mathematical Foundations of Computer Science, MFCS 2006, Stará Lesná, Slovakia, August 28-September 1, 2006, Proceedings*, volume 4162 of *Lecture Notes in Computer Science*, pages 267–279. Springer, 2006. doi:10.1007/11821069_24.

15 Josep Díaz and Marcin Kaminski. MAX-CUT and MAX-BISECTION are NP-hard on unit disk graphs. *Theor. Comput. Sci.*, 377(1-3):271–276, 2007. doi:10.1016/j.tcs.2007.02.013.

16 Florent Foucaud, George B. Mertzios, Reza Naserasr, Aline Parreau, and Petru Valicov. Identification, location-domination and metric dimension on interval and permutation graphs. II. Algorithms and complexity. *Algorithmica*, 78(3):914–944, 2017. doi:10.1007/s00453-016-0184-1.

17 Michael R Garey and David S Johnson. A guide to the theory of NP-completeness. *Computers and Intractability*, pages 37–79, 1990.

18 Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, 1995. doi:10.1145/227683.227684.

19 Martin Charles Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Elsevier, 2004.

20 Venkatesan Guruswami. Maximum cut on line and total graphs. *Discret. Appl. Math.*, 92(2-3):217–221, 1999. doi:10.1016/S0166-218X(99)00056-6.

21 F. Hadlock. Finding a maximum cut of a planar graph in polynomial time. *SIAM J. Comput.*, 4(3):221–225, 1975. doi:10.1137/0204019.

22 Hiroshi Imai and Takao Asano. Finding the connected components and a maximum clique of an intersection graph of rectangles in the plane. *Journal of Algorithms*, 4(4):310–323, 1983. doi:10.1016/0196-6774(83)90012-3.

23 David S Johnson. The NP-completeness column: an ongoing guide. *Journal of Algorithms*, 6(3):434–451, 1985. doi:10.1016/0196-6774(85)90012-4.

24 John R Jungck, Gregg Dick, and Amy Gleason Dick. Computer-assisted sequencing, interval graphs, and molecular evolution. *Biosystems*, 15(3):259–273, 1982. doi:10.1016/0303-2647(82)90010-7.

25 Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher, editors, *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972. doi:10.1007/978-1-4684-2001-2_9.

26 J. Mark Keil. Finding hamiltonian circuits in interval graphs. *Inf. Process. Lett.*, 20(4):201–206, 1985. doi:10.1016/0020-0190(85)90050-X.

27 Subhash Khot. On the power of unique 2-prover 1-round games. In John H. Reif, editor, *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 767–775. ACM, 2002. doi:10.1145/509907.510017.

28 Jan Kratochvíl, Tomás Masarík, and Jana Novotná. U-bubble model for mixed unit interval graphs and its applications: The maxcut problem revisited. In Javier Esparza and Daniel Král', editors, *45th International Symposium on Mathematical Foundations of Computer Science, MFCS 2020, August 24-28, 2020, Prague, Czech Republic*, volume 170 of *LIPIcs*, pages 57:1–57:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.MFCS.2020.57.

29 Chin Lung Lu and Chuan Yi Tang. A linear-time algorithm for the weighted feedback vertex problem on interval graphs. *Inf. Process. Lett.*, 61(2):107–111, 1997. doi:10.1016/S0020-0190(96)00193-7.

30 Madhav V. Marathe, R. Ravi, and C. Pandu Rangan. Generalized vertex covering in interval graphs. *Discret. Appl. Math.*, 39(1):87–93, 1992. doi:10.1016/0166-218X(92)90116-R.

31 Dániel Marx. A short proof of the NP-completeness of minimum sum interval coloring. *Oper. Res. Lett.*, 33(4):382–384, 2005. doi:10.1016/j.orl.2004.07.006.

32 Peisen Zhang, Eric A Schon, Stuart G Fischer, Eftihia Cayanis, Janie Weiss, Susan Kistler, and Philip E Bourne. An algorithm based on graph theory for the assembly of contigs in physical mapping of DNA. *Bioinformatics*, 10(3):309–317, 1994. doi:10.1093/bioinformatics/10.3.309.

Lower Bounds for Semialgebraic Range Searching and Stabbing Problems

Peyman Afshani ✉
Aarhus University, Denmark

Pingan Cheng ✉
Aarhus University, Denmark

Abstract

In the semialgebraic range searching problem, we are given a set of n points in $\mathbb{R}^d$ and we want to preprocess the points such that for any query range belonging to a family of constant complexity semialgebraic sets (Tarski cells), all the points intersecting the range can be reported or counted efficiently. When the ranges are composed of simplices, then the problem is well-understood: it can be solved using $S(n)$ space and with $Q(n)$ query time with $S(n)Q^d(n) = \tilde{O}(n^d)$ where the $\tilde{O}(\cdot)$ notation hides polylogarithmic factors and this trade-off is tight (up to $n^{o(1)}$ factors). Consequently, there exists "low space" structures that use $O(n)$ space with $O(n^{1-1/d})$ query time and "fast query" structures that use $O(n^d)$ space with $O(\log^{d+1} n)$ query time. However, for the general semialgebraic ranges, only "low space" solutions are known, but the best solutions match the same trade-off curve as the simplex queries, with $O(n)$ space and $\tilde{O}(n^{1-1/d})$ query time. It has been conjectured that the same could be done for the "fast query" case but this open problem has stayed unresolved.

Here, we disprove this conjecture. We give the first nontrivial lower bounds for semilagebraic range searching and other related problems. More precisely, we show that any data structure for reporting the points between two concentric circles, a problem that we call 2D annulus reporting problem, with $Q(n)$ query time must use $S(n) = \overset{o}{\Omega}(n^3/Q(n)^5)$ space where the $\overset{o}{\Omega}(\cdot)$ notation hides $n^{o(1)}$ factors, meaning, for $Q(n) = O(\log^{O(1)} n)$, $\overset{o}{\Omega}(n^3)$ space must be used. In addition, we study the problem of reporting the subset of input points between two polynomials of the form $Y = \sum_{i=0}^{\Delta} a_i X^i$ where values $a_0, \cdots, a_\Delta$ are given at the query time, a problem that we call polynomial slab reporting. For this, we show a space lower bound of $\overset{o}{\Omega}(n^{\Delta+1}/Q(n)^{\Delta^2+\Delta})$, which shows for $Q(n) = O(\log^{O(1)} n)$, we must use $\overset{o}{\Omega}(n^{\Delta+1})$ space. We also consider the dual problems of semialgebraic range searching, semialgebraic stabbing problems, and present lower bounds for them. In particular, we show that in linear space, any data structure that solves 2D annulus stabbing problems must use $\Omega(n^{2/3})$ query time. Note that this almost matches the upper bound obtained by lifting 2D annuli to 3D. Like semialgebraic range searching, we also present lower bounds for general semialgebraic slab stabbing problems. Again, our lower bounds are almost tight for linear size data structures in this case.

2012 ACM Subject Classification Theory of computation → Computational geometry

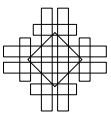
Keywords and phrases Computational Geometry, Data Structures and Algorithms

Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.8

Related Version *Full Version*: `https://arxiv.org/abs/2012.00704`

Funding Supported by DFF (Det Frie Forskningsråd) of Danish Council for Independent Research under grant ID DFF–7014–00404.

Acknowledgements The authors would like to thank Esther Ezra for sparking the initial ideas behind the proof.



1 Introduction

We address one of the biggest open problems of the recent years in the range searching area. Our main results are lower bounds in the pointer machine model of computation that essentially show that the so-called "fast query" version of the semialgebraic range reporting problem is "impervious" to the algebraic techniques. Our main result reveals that to obtain polylogarithmic query time, the data structure requires $\mathring{\Omega}(n^{\Delta+1})$ space[1], where the constant depends on Δ, n is the input size, and $\Delta+1$ is the number of parameters of each "polynomial inequality" (these will be defined more clearly later). Thus, we refute a relatively popular recent conjecture that data structures with $\mathring{O}(n^d)$ space and polylogarithmic query time could exist, where d is the dimension of the input points. Surprisingly, the proofs behind these lower bounds are simple, and these lower bounds could have been discovered years ago as the tools we use already existed decades ago.

Range searching is a broad area of research in which we are given a set P of n points in $\mathbb{R}^d$ and the goal is to preprocess P such that given a query range $\mathcal{R}$, we can count or report the subset of P that lies in $\mathcal{R}$. Often $\mathcal{R}$ is restricted to a fixed family of ranges, e.g., in simplex range counting problem, $\mathcal{R}$ is a simplex in $\mathbb{R}^d$ and the goal is to report $|P \cap \mathcal{R}|$, or in halfspace range reporting problem, $\mathcal{R}$ is a halfspace and the goal is to report $P \cap \mathcal{R}$. Range searching problems have been studied extensively and they have numerous variants. For an overview of this topic, we refer the readers to an excellent survey by Agarwal [17].

Another highly related problem which can be viewed as the "dual" of this problem is range stabbing: we are given a set R of ranges as input and the goal is to preprocess R such that given a query point p, we can count or report the ranges of R containing p efficiently. Here, we focus on the reporting version of range stabbing problems.

1.1 Range Searching: A Very Brief Survey

1.1.1 Simplex Range Searching

Simplices is one of the most fundamental family of queries. In fact, if the query is decomposable (such as range counting or range reporting queries), then simplices can be used as "building blocks" to answer more complicated queries: for a query $\mathcal{R}$ which is a polyhedral region of $O(1)$ complexity, we can decompose it into $O(1)$ disjoint simplices (with a constant that depends on d) and thus answering $\mathcal{R}$ can be reduced to answering $O(1)$ simplicial queries.

Simplicial queries were hotly investigated in 1980s and this led to development of two important tools in computational geometry: cuttings and partition theorem and both of them have found applications in areas not related to range searching.

1.1.1.1 Cuttings and Fast Data Structures

"Fast query" data structures can answer simplex range counting or reporting queries in polylogarithmic query time but by using $O(n^d)$ space and they can be built using cuttings. In a nut-shell, given a set H of n hyperplanes in $\mathbb{R}^d$, a $\frac{1}{r}$-cutting, is a decomposition of $\mathbb{R}^d$ into $O(r^d)$ simplices such that each simplex is intersected by $O(n/r)$ hyperplanes of H. These were developed by some of the pioneers in the range searching area, such as Clarkson [15], Haussler and Welzl [19], Chazelle and Friedman [8], Matoušek [20], finally culminating in a

[1] $\mathring{\Omega}(\cdot)$, $\mathring{O}(\cdot)$, $\mathring{\Theta}(\cdot)$ notations hide $n^{o(1)}$ factors and $\tilde{\Omega}(\cdot)$, $\tilde{O}(\cdot)$, $\tilde{\Theta}(\cdot)$ notations hide $\log^{O(1)} n$ factors.

result of Chazelle [11] who optimized various aspects of cuttings. Using cuttings, one can answer simplex range counting, or reporting queries with $O(n^d)$ space and $O((\log n)^{d+1} + k)$ query time (where k is the output size) [21]. The query time can be lowered to $O(\log n)$ by increasing the space slightly to $O(n^{d+\varepsilon})$ for any constant $\varepsilon > 0$ [14]. An interested reader can refer to a book on cuttings by Chazelle [12].

1.1.1.2 The Partition Theorem and Space-efficient Data Structures

At the opposite end of the spectrum, simplex range counting or reporting queries can be answered using linear space but with higher query time of $O(n^{1-1/d})$, using partition trees and the related techniques. This branch of techniques has a very interesting history. In 1982, Willard [24] cleverly used ham sandwich theorem to obtain a linear-sized data structure with query time of $O(n^\gamma)$ for some constant $\gamma < 1$ for simplicial queries in 2D. After a number of attempts that either improved the exponent or generalized the technique to higher dimensions, Welzl [23] in 1982 provided the first optimal exponent for the partition trees, then Chazelle et al. [14] provided the first near-linear size data structure with query time of roughly $O(n^{1-1/d})$. Finally, a data structure with $O(n)$ space and $O(n^{1-1/d})$ query time was given by Matoušek [21]. This was also simplified recently by Chan [7].

1.1.1.3 Space/Query Time Trade-off

It is possible to combine fast query data structures and linear-sized data structures to solve simplex queries with $S(n)$ space and $Q(n)$ query time such that $S(n)Q(n)^d = \tilde{O}(n^d)$. This trade-off between space and query time is optimal, at least in the pointer machine model and in the semigroup model [1, 13, 9].

1.1.1.4 Multi-level Structures, Stabbing and Other Related Queries

By using multi-level data structures, one can solve more complicated problems where both the input and the query shapes can be simplicial objects of constant complexity. The best multi-level data structures use one extra $\log n$ factor in space and query time per level [7] and there exist lower bounds that show space/query time trade-off should blow up by at least $\log n$ factor per level [2]. This means that problems such as simplex stabbing (where the input is a set of simplices and we want to output the simplices containing a given query point) or simplex-simplex containment problem (where the input is a set of simplices, and we want to output simplices fully contained in a query simplex) all have the same trade-off curve of $S(n)Q(n)^d = \tilde{O}(n^d)$ between space $S(n)$ and query time $Q(n)$.

Thus, one can see that the simplex range searching as well as its generalization to problems where both the input and the query ranges are "flat" objects is very well understood. However, there are many natural query ranges that cannot be represented using simplices, e.g., when query ranges are spheres in $\mathbb{R}^d$. This takes us to semialgebraic range searching.

1.1.2 Semialgebraic Range Searching

A semialgebraic set is defined as a subset of $\mathbb{R}^d$ that can be described as the union or intersection of $O(1)$ ranges, where each range is defined by d-variate polynomial inequality of degree at most Δ, defined by at most B values given at the query time; we call B the *parametric dimension*. For instance, with $B = 3$, $\Delta = 2$, and given three values a, b and c at the query time, a circular query can be represented as $\{(X, Y) \in \mathbb{R}^2 | (X - a)^2 + (Y - b)^2 \leq c^2\}$. In semialgebraic range searching, the queries are semialgebraic sets.

Before the recent "polynomial method revolution", the tools available to deal with semialgebraic range searching were limited, at least compared to the simplex queries. One way to deal with semialgebraic range searching is through linearization [25]. This idea maps the input points to $\mathbb{R}^L$, for some potentially large parameter L, such that each polynomial inequality can be represented as a halfspace. Consequently, semialgebraic range searching can be solved with the space/query time trade off of $S(n)Q(n)^L = \tilde{O}(n^L)$. The exponent of $Q(n)$ in the trade-off can be improved (increased) a bit by exploiting that in $\mathbb{R}^L$, the input set actually lies in a d-dimensional surface [5]. It is also possible to build "fast query" data structures but using $O(n^{2B-4+\varepsilon})$, but only in specific cases [5] (see [17] for details).

In 2009, Zeev Dvir [16] proved the discrete Kakeya problem with a very elegant and simple proof, using a polynomial method. Within a few years, this led to revolution in discrete and computational geometry, one that was ushered in by Katz and Guth's almost tight bound on Erdős distinct distances problem [18]. For a while, the polynomial method did not have much algorithmic consequences but this changed with the work of Agarwal, Matoušek, and Sharir [6] where they showed that at least as long as linear-space data structures are considered, semialgebraic range queries can essentially be solved within the same time as simplex queries (ignoring some lower order terms). Later developments (and simplifications) of their approach by Matoušek and Patáková [22] lead to the current best results: a data structure with linear size and with query time of $\tilde{O}(n^{1-1/d})$.

1.1.2.1 Fast Queries for Semialgebraic Range Searching: an Open Problem

Nonetheless, despite the breakthrough results brought on by the algebraic techniques, the fast query case still remained unsolved, even in the plane: e.g., the best known data structures for answering circular queries with polylogarithmic query time still use $\tilde{O}(n^3)$ space, by using linearization to $\mathbb{R}^3$. The fast query case of semialgebraic range searching has been explicitly mentioned as a major open problem in multiple recent publications[2]. In light of the breakthrough result of Agarwal et al. [6], it is quite reasonable to conjecture that semialgebraic range searching should have the same trade-off curve of $S(n)Q(n)^d = \tilde{O}(n^d)$.

Nonetheless, the algebraic techniques have failed to make sufficient advances to settle this open problem. The best known result is given recently by Agarwal et al. [4]. They showed it is possible to build "fast query" semialgebraic range searching data structures using $O(n^{B+\epsilon})$ space. In general, B can be much larger than d and thus it leaves a big gap between current best upper bound and the conjectured one. Given that it took a revolution caused by the polynomial method to advance our knowledge of the "low space" case of semialgebraic range searching, it is not too outrageous to imagine that perhaps equally revolutionary techniques are needed to settle the "fast query" case of semialgebraic range searching.

1.1.3 Semialgebraic Range Stabbing

Another important problem is semialgebraic stabbing, where the input is a set of n semialgebraic sets, i.e., "ranges", and queries are points. The goal is to output the input ranges that contain a query point. Here, sometimes "fast query" data structures are possible, for example by observing that an arrangements of n disks in the plane has $O(n^2)$ complexity

[2] To quote Agarwal et al. [6],"[a] very interesting and challenging problem is, in our opinion, the fast-query case of range searching with constant-complexity semialgebraic sets, where the goal is to answer a query in $O(\log n)$ time using roughly $O(n^d)$ space." The same conjecture is repeated in a different survey [3] and it is also emphasized that the question is even open for disks in the plane, "... whether a disk range-counting query in $\mathbb{R}^2$ be answered in $O(\log n)$ time using $O(n^2)$ space?".

and thus counting or reporting the disks stabbed by a query point can be done with $O(n^2)$ space and $O(\log n)$ query time. However, it seems difficult to make advancements in the "low space" side of things; the only known data structure with $O(n)$ space is one that uses linearization to 3D that results in $\tilde{O}(n^{2/3})$ query time.

1.2 Our Results

Our main results are lower bounds in the pointer machine model of computation for four central problems defined below. In the *2D polynomial slab reporting* problem, given a set $\mathcal{P}$ of n points in $\mathbb{R}^2$, the task is to preprocess $\mathcal{P}$ such that given a query 2D polynomial slab $\mathcal{R}$, the points contained in the polynomial slab, i.e., $\mathcal{R} \cap \mathcal{P}$, can be reported efficiently. Informally, a 2D polynomial slab is the set of points (x, y) such that $P(x) \leq y \leq P(x) + w$, for some univariate polynomial $P(x)$ of degree Δ and value w given at the query time. In the *2D polynomial slab stabbing* problem, the input is a set of n polynomial slabs and the query is a point q and the goal is to report all the slabs that contain q. Similarly, in the *2D annulus reporting* problem, the input is a set P of n points in $\mathbb{R}^2$ and the query is a "annulus", the region between two concentric circles. Finally, in *2D annulus stabbing* problem, the input is a set of n annuli, the query is a point q and the goal is to report all the annuli that contain q.

For polynomial slab queries, we show that if a data structure answers queries in $Q(n)+O(k)$ time, where k is the output size, using $S(n)$ space, then $S(n) = \mathring{\Omega}(n^{\Delta+1}/Q(n)^{\Delta^2+\Delta})$; the hidden constants depend on Δ. So for "fast queries", i.e., $Q(n) = \tilde{O}(1)$, $\mathring{\Omega}(n^{\Delta+1})$ space must be used. This is *almost tight* as the exponent matches the upper bounds obtained by linearization as well as the recent upper bound of Agarwal et al. [4]! Also, we prove that any structure that answers polynomial slab reporting queries in $Q(n) + O(k)$ time must use $\Omega(n^{1+1/\Delta}/Q(n)^{(\Delta+1)/\Delta^2})$ space. In the "low space" setting, when $S(n) = O(n)$, this gives $Q(n) = \Omega(n^{1-1/(\Delta+1)})$. This is once again *almost tight*, as it matches the upper bounds obtained by linearization for when $S(n) = \tilde{O}(n)$.

For the annulus reporting problem, our bound sharpens to $S(n) = \mathring{\Omega}(n^3/Q(n)^5)$. For the annulus stabbing problem, we show $S(n) = \Omega(n^{3/2}/Q(n)^{3/4})$, e.g., in "low space" setting when $S(n) = O(n)$, we must have $Q(n) = \Omega(n^{2/3})$; compare this with simplex stabbing queries can be solved with $O(n)$ space and $\tilde{O}(\sqrt{n})$ query time. As before, this is almost tight, as it matches the upper bounds obtained by linearization to 3D for when $S(n) = \tilde{O}(n)$.

Somewhat disappointedly, no revolutionary new technique is required to obtain these results. We use novel ideas in the construction of "hard input instances" but otherwise we use the two widely used pointer machine lower bound frameworks by Chazelle [10], Chazelle and Rosenberg [13], and Afshani [1]. Our results are summarized in Table 1.

2 Preliminaries

We first review the related geometric reporting data structure lower bound frameworks. The model of computation we consider is (an augmented version of) the pointer machine model.

In this model, the data structure is a directed graph M. Let $\mathcal{S}$ be the set of input elements. Each cell of M stores an element of $\mathcal{S}$ and two pointers to other cells. Assume a query q requires a subset $\mathcal{S}_q \subset \mathcal{S}$ to be output. For the query, we only charge for the pointer navigations. Let M_q be the smallest connected subgraph, s.t., every element of $\mathcal{S}_q$ is stored in at least one element of M_q. Clearly, $|M|$ is a lower bound for space and $|M_q|$ is a lower bound for query time. Note that this grants the algorithm unlimited computational power as well as full information about the structure of M.

In this model, there are two main lower bound frameworks, one for range reporting [10, 13], and the other for its dual, range stabbing [1]. We describe them in detail here.

2.1 A Lower Bound Framework for Range Reporting Problems

The following result by Chazelle [10] and later Chazelle and Rosenberg [13] provides a general lower bound framework for range reporting problems. In the problem, we are given a set $\mathcal{S}$ of n points in $\mathbb{R}^d$ and the queries are from a set $\mathcal{R}$ of ranges. The task is to build a data structure such that given any query range $\mathcal{R} \in \mathcal{R}$, we can report the points intersecting the range, i.e., $\mathcal{R} \cap \mathcal{S}$, efficiently.

▶ **Theorem 1** (Chazelle [10] and Chazelle and Rosenberg [13]). *Suppose there is a data structure for range reporting problems that uses at most $S(n)$ space and can answer any query in $Q(n) + O(k)$ time where n is the input size and k is the output size. Assume we can show that there exists an input set $\mathcal{S}$ of n points satisfying the following: There exist m subsets $q_1, q_2, \cdots, q_m \subset \mathcal{S}$, where $q_i, i = 1, \cdots, m$, is the output of some query and they satisfy the following two conditions: (i) for all $i = 1, \cdots, m$, $|q_i| \geq Q(n)$; and (ii) the size of the intersection of every α distinct subsets $q_{i_1}, q_{i_2}, \cdots, q_{i_\alpha}$ is bounded by some value $c \geq 2$, i.e., $|q_{i_1} \cap q_{i_2} \cap \cdots \cap q_{i_\alpha}| \leq c$. Then $S(n) = \Omega(\frac{\sum_{i=1}^{m} |q_i|}{\alpha 2^{O(c)}}) = \Omega(\frac{mQ(n)}{\alpha 2^{O(c)}})$.*

To use this framework, we need to exploit the property of the considered problem and come up with a construction that satisfies the two conditions above. Often, the construction is randomized and thus one challenge is to satisfy condition (ii) in the worst-case. This can be done by showing that the probability that (ii) is violated is very small and then using a union bound to prove that with positive probability the construction satisfies (ii) in the worst-case.

Table 1 Our Results, $*$ indicates this paper. In the table, $\mathring{\Omega}(\cdot)$ and $\mathring{O}(\cdot)$ notations hide $n^{o(1)}$ factors, and $\tilde{O}(\cdot)$ notation hides $\log^{O(1)} n$ factors.

Problem	Lower Bound	Upper Bound
2D Polynomial Slab Reporting	$S(n) = \mathring{\Omega}\left(\frac{n^{\Delta+1}}{Q(n)^{\Delta^2+\Delta}}\right)^*$	$S(n) = \tilde{O}\left(\frac{n^{\Delta+1}}{Q(n)^{2\Delta}}\right)$ [5, 6, 21]
When $Q(n) = \mathring{O}(1)$	$\boldsymbol{S(n) = \mathring{\Omega}\left(n^{\Delta+1}\right)^*}$	$\boldsymbol{S(n) = \mathring{O}\left(n^{\Delta+1}\right)}$ [5, 6, 21]
2D Annulus Reporting	$S(n) = \mathring{\Omega}\left(\frac{n^3}{Q(n)^5}\right)^*$	$S(n) = \tilde{O}\left(\frac{n^3}{Q(n)^4}\right)$ [5, 6, 21]
When $Q(n) = \mathring{O}(1)$	$\boldsymbol{S(n) = \mathring{\Omega}\left(n^3\right)^*}$	$\boldsymbol{S(n) = \mathring{O}\left(n^3\right)}$ [5, 6, 21]
2D Polynomial Slab Stabbing	$S(n) = \Omega\left(\frac{n^{1+1/\Delta}}{Q(n)^{(\Delta+1)/\Delta^2}}\right)^*$	$S(n) = \tilde{O}\left(\frac{n^2}{Q(n)^{(\Delta+1)/\Delta}}\right)^a$
When $S(n) = \mathring{O}(n)$	$\boldsymbol{Q(n) = \mathring{\Omega}\left(n^{1-1/(\Delta+1)}\right)^*}$	$\boldsymbol{S(n) = \mathring{O}\left(n^{1-1/(\Delta+1)}\right)}$
2D Annulus Stabbing	$S(n) = \Omega\left(\frac{n^{3/2}}{Q(n)^{3/4}}\right)^*$	$S(n) = \tilde{O}\left(\frac{n^2}{Q(n)^{3/2}}\right)^b$
When $S(n) = \mathring{O}(n)$	$\boldsymbol{Q(n) = \mathring{\Omega}\left(n^{2/3}\right)^*}$	$\boldsymbol{Q(n) = \mathring{O}\left(n^{2/3}\right)}$

[a] The subdivision formed by n degree Δ polynomial slabs has complexity $O(n^2)$ (for some constant depending on Δ). We partition the subdivision into vertical strips where for any strip any slab intersecting it fully span the strip and the number of slab changes of adjacent strips is $O(1)$. Consider these strips from left to right, we are solving a special dynamic slab stabbing problem. We can solve this problem by building a persistent interval tree using $O(n^2)$ space that answers each query in time $O(\log n + k)$. On the other hand, we can solve the problem in $O(n)$ space and $O(n^{1-1/(\Delta+1)} + k)$ time by linearization. Combining these two solutions using [21] gives the tradeoff.

[b] Similar to 2D polynomial slab stabbing.

2.2 A Lower Bound Framework for Range Stabbing Problems

Range stabbing problems can be viewed as the dual of range reporting problems. In this problem, we are given a set $\mathcal{R}$ of n ranges, and the queries are from a set $\mathcal{Q}$ of n points. The task is to build a data structure such that given any query point $q \in \mathcal{Q}$, we can report the ranges "stabbed" by this query point, i.e., $\{\mathcal{R} \in \mathcal{R} : \mathcal{R} \cap q \neq \emptyset\}$, efficiently. A recent framework by Afshani [1] provides a simple way to get the lower bound of such problems.

▶ **Theorem 2** (Afshani [1]). *Suppose there is a data structure for range stabbing problems that uses at most $S(n)$ space and can answer any query in $Q(n) + O(k)$ time where n is the input size and k is the output size. Assume we can show that there exists an input set $R \subset \mathcal{R}$ of n ranges that satisfy the following: (i) every query point of the unit square U is contained in at least $t \geq Q(n)$ ranges; and (ii) the area of the intersection of every $\alpha < t$ ranges is at most v. Then $S(n) = \Omega(\frac{t}{v2^{O(\alpha)}}) = \Omega(\frac{Q(n)}{v2^{O(\alpha)}})$.*

This is very similar to framework of Theorem 1 but often it requires no derandomization.

3 2D Polynomial Slab Reporting and Stabbing

We first consider the case when query ranges are 2D polynomial slabs. The formal definition of 2D polynomial slabs is as follows.

▶ **Definition 3.** *Let $P(x) = \sum_{i=0}^{\Delta} a_i x^i$, where $a_\Delta \neq 0$, be a degree Δ univariate polynomial. A 2D polynomial slab is a pair $(P(x), w)$, where $P(x)$ is called the base polynomial and $w > 0$ the width of the polynomial slab. The polynomial slab is then defined as $\{(x, y) \in \mathbb{R}^2 : P(x) \leq y \leq P(x) + w\}$.*

3.1 2D Polynomial Slab Reporting

We consider the 2D polynomial slab reporting problem in this section, where the input is a set $\mathcal{P}$ of n points in $\mathbb{R}^2$, and the query is a polynomial slab. This is an instance of semialgebraic range searching where we have two polynomial inequalities where each inequality has degree Δ and it is defined by $\Delta+1$ parameters given at the query time (thus, $B = \Delta+1$). Note that $\Delta+1$ is also the dimension of linearization for this problem, meaning, the 2D polynomial slab reporting problem can be lifted to the simplex range reporting problem in $\mathbb{R}^{\Delta+1}$. Our main result shows that for fast queries (i.e., when the query time is polylogarithmic), this is tight, by showing an $\mathring{\Omega}(n^{\Delta+1})$ space lower bound, in the pointer machine model of computation.

To do that, we will use Chazelle's framework. In our construction of a hard input instance, a derandomization process will be needed. We do this using the following two general lemmas. For the proofs of these lemmas, see the full version of the paper.

▶ **Lemma 4.** *Let $\mathcal{P}$ be a set of n points chosen uniformly at random in a square S of side length n in $\mathbb{R}^2$. Let $\mathcal{R}$ be a set of ranges in S such that (i) the intersection area of any $t \geq 2$ ranges $\mathcal{R}_1, \mathcal{R}_2, \cdots \mathcal{R}_t \in \mathcal{R}$ is bounded by $O\left(n/2^{\sqrt{\log n}}\right)$; (ii) the total number of intersections is bounded by $O\left(n^{2k}\right)$ for $k \geq 1$. Then with probability $> \frac{1}{2}$, for all distinct ranges $\mathcal{R}_1, \mathcal{R}_2, \cdots, \mathcal{R}_t \in \mathcal{R}$, $|\mathcal{R}_1 \cap \mathcal{R}_2 \cap \cdots \mathcal{R}_t \cap \mathcal{P}| < 3k\sqrt{\log n}$.*

▶ **Lemma 5.** *Let $\mathcal{P}$ be a set of n points chosen uniformly at random in a square S of side length n in $\mathbb{R}^2$. Let $\mathcal{R}$ be a set of ranges in S such that (i) the intersection area of any range $\mathcal{R} \in \mathcal{R}$ and S is at least cnt for some constant $c \geq 4k$ and a parameter $t \geq \log n$, where $k \geq 2$; (ii) the total number of ranges is bounded by $O\left(n^{k+1}\right)$. Then with probability $> \frac{1}{2}$, for every range $\mathcal{R} \in \mathcal{R}$, $|\mathcal{R} \cap \mathcal{P}| \geq t$.*

Given a univariate polynomial $P(x)$, the following simple lemma establishes the relationship between the coefficient of the maximum degree term and the maximum range within which its value is bounded. This lemma will be used to upper bound the intersection area of two polynomial slabs. For the proof of this lemma, see the full version of the paper.

▶ **Lemma 6.** *Let $P(x) = \sum_{i=0}^{\Delta} a_i x^i$ be a degree Δ univariate polynomial where $\Delta > 0$ and $|a_\Delta| \geq d$ for some positive d. Let w be any positive value and x_l be a parameter. If $|P(x)| \leq w$ for all $x \in [x_l, x_l + t]$, then $t \leq (\Delta+1)^3 \left(\frac{w}{d}\right)^{\frac{1}{\Delta}}$.*

With Lemma 6 at hand, we now show a lower bound for polynomial slab reporting.

▶ **Theorem 7.** *Let $\mathcal{P}$ be a set of n points in $\mathbb{R}^2$. Let $\mathcal{R}$ be the set of all 2D polynomial slabs $\{(P(x), w) : \deg(P) = \Delta \geq 2, w > 0\}$. Then any data structure for $\mathcal{P}$ that solves polynomial slab reporting for queries from $\mathcal{R}$ with query time $Q(n) + O(k)$, where k is the output size, uses $S(n) = \mathring{\Omega}\left(n^{\Delta+1}/Q(n)^{\Delta^2+\Delta}\right)$ space.*

Proof. We use Chazelle's framework to prove this theorem. To this end, we will need to show the existence of a hard input instance. We do this as follows. In a square S, we construct a set of special polynomial slabs with the following properties: (i) The intersection area of any two slabs is small; and (ii) The area of each slab inside S is relatively large. Intuitively and consequently, if we sample n points uniformly at random in S, in expectation, few points will be in the intersection of two slabs, and many points will be in each slab. Intuitively, this satisfies the two conditions of Theorem 1. By picking parameters carefully and a derandomization process, we get our theorem. Next, we describe the details.

Consider a square $S = [0, n] \times [0, n]$. Let d, w be some parameters to be specified later. We generate a set of $\Theta\left(\left(\frac{n}{2d}\right)^{\Delta} \cdot \frac{n}{w}\right)$ polynomial slabs $(P(x), w)$ with

$$P(x) = \left(\sum_{i=1}^{\Delta} \frac{j_i d x^i}{n^i}\right) + kw$$

where $j_i = \lfloor \frac{n}{2d} \rfloor, \lfloor \frac{n}{2d} \rfloor + 1, \cdots, \lfloor \frac{n}{d} \rfloor$ for $1 \leq i \leq \Delta$ and $k = \lfloor \frac{n}{4w} \rfloor, \lfloor \frac{n}{4w} \rfloor + 1, \cdots, \lfloor \frac{n}{2w} \rfloor$. Note that we normalize the coefficients such that for any polynomial slab in range $x \in [0, n]$, a quarter of this slab is contained in S if $w < n/6$. To show this, it is sufficient to show that every polynomial is inside S, for every $x \in [0, n/4]$. As all the coefficients of the polynomials are positive, it is sufficient to upper bound $P(n/4)$, among all the polynomials $P(x)$ that we have generated. Similarly, this maximum is attained when all the coefficients are set to their maximum value, i.e., when $j_i = n/d$ and $k = n/(2w)$, resulting in the polynomial $P_u(x) = \left(\sum_{i=1}^{\Delta} x^i / n^{i-1}\right) + \frac{n}{2}$. Now it easily follows that $P_u(n/4) < 5n/6$. Then, the claim follows from the following simple observation.

▶ **Observation 8.** *The area of a polynomial slab $\{P(x), w\}$ for when $a \leq x \leq b$ is $(b-a)w$.*

Proof. The claimed area is $(\int_a^b (P(x) + w)dx) - (\int_a^b P(x)dx) = \int_a^b w dx = (b-a)w$. ◀

Next, we bound the area of the intersection of two polynomial slabs. Consider two distinct slabs $\mathcal{R}_p = (P(x), w)$ and $\mathcal{R}_q = (Q(x), w)$. Observe that by our construction, if $P(x)$ and $Q(x)$ only differ in their constant terms, their intersection is empty. So we only consider the case that there exists some $0 < i \leq \Delta$, such that the coefficients for x^i are different in $P(x)$ and $Q(x)$. As each slab is created using two polynomials of degree Δ, $\mathcal{R}_q \cap \mathcal{R}_p$ can have at most $O(\Delta)$ connected regions. Consider one connected region $\mathfrak{R}$ and let the

interval $\eta = [x_1, x_2] \subset [0, n]$, be the projection of $\mathfrak{R}$ onto the X-axis. Define the polynomial $R(x) = P(x) - Q(x)$ and observe that we must have $|R(x)| \leq w$ for all $x \in [x_1, x_2]$. We now consider the coefficient of the highest degree term of $R(x)$. Let $j_i d/n^i$ (resp. $j'_i d/n^i$) be the coefficient of the degree i term in $P(x)$ (resp. $Q(x)$). Clearly, if $j_i = j'_i$, then the coefficient of x^i in $R(x)$ will be zero. Thus, to find the highest degree term in $R(x)$, we need to consider the largest index i such that $j_i \neq j'_i$; in this case, $R(x)$ will have degree i and coefficient of x_i will have absolute value $\left|(j_i - j'_i)d/n^i\right| \geq d/n^i$. When $w \leq d$, by Lemma 6, $x_2 - x_1 \leq O(\Delta^3)\left(\frac{wn^i}{d}\right)^{1/i} \leq O(\Delta^3)n\left(\frac{w}{d}\right)^{1/\Delta}$. Next, by Observation 8, the area of the intersection of $\mathcal{R}_q$ and $\mathcal{R}_p$ is $O(\Delta^3)nw\left(\frac{w}{d}\right)^{1/\Delta}$.

We pick $d = c\Delta^{3\Delta}w^{\Delta+1}2^{\Delta\sqrt{\log n}}$ and $w = 16\Delta Q(n)$, for a large enough constant c. Then, the intersection area of any two polynomial slabs is bounded by $n/2^{\sqrt{\log n}}$. Since in total we have generated $O(n^{\Delta+1})$ slabs, the total number of intersections they can form is bounded by $O(n^{2(\Delta+1)})$. By Lemma 4, with probability $> \frac{1}{2}$, the number of points of $\mathcal{P}$ in any intersection of two polynomial slabs is at most $3(\Delta+1)\sqrt{\log n}$. Also, as we have shown that the intersection area of every slab with S is at least $nw/4 = 4\Delta nQ(n)$, by Lemma 5, with probability more than $\frac{1}{2}$, each polynomial slab has at least $Q(n)$ points of $\mathcal{P}$.

It thus follows that with positive probability, both conditions of Theorem 1 are satisfied, and consequently, we obtain the lower bound of

$$S(n) = \Omega\left(\frac{Q(n)\cdot\left(\frac{n}{2d}\right)^{\Delta}\cdot\frac{n}{w}}{2^{3(\Delta+1)\sqrt{\log n}}}\right) = \overset{o}{\Omega}\left(\frac{n^{\Delta+1}}{Q(n)^{\Delta^2+\Delta}}\right). \qquad \blacktriangleleft$$

So for the "fast query" case data structure, by picking $Q(n) = O(\log^{O(1)} n)$, we obtain a space lower bound of $S(n) = \overset{o}{\Omega}(n^{\Delta+1})$.

3.2 2D Polynomial Slab Stabbing

By small modifications, our construction can also be applied to obtain a lower bound for (the reporting version of) polynomial slab stabbing problems using Theorem 2.

One modification is that we need to generate the slabs in such a way that they cover the entire square S. The framework provided through Theorem 2 is more stream-lined and derandomization is not needed and we can directly apply the "volume upper bound" obtained through Lemma 6. There is also no $n^{o(1)}$ factor loss (our lower bound actually uses $\Omega(\cdot)$ notation). The major change is that we need to use different parameters since we need to create n polygonal slabs, as now they are the input. For the details refer to the full version of the paper.

▶ **Theorem 9.** *Give a set $\mathcal{R}$ of n 2D polynomial slabs $\{(P(x), w) : deg(P) = \Delta \geq 2, w > 0\}$, any data structure for $\mathcal{R}$ solving the 2D polynomial slab stabbing problem with query time $Q(n) + O(k)$ uses $S(n) = \Omega\left(\frac{n^{1+\frac{1}{\Delta}}}{Q(n)^{\frac{\Delta+1}{\Delta^2}}}\right)$ space, where k is the output size.*

So for any data structure that solves the 2D polynomial slab stabbing problem using $S(n) = O(n)$ space, Theorem 9 implies that its query time must be $Q(n) = \Omega(n^{1-1/(\Delta+1)})$.

4 2D Annulus Reporting and Stabbing

4.1 2D Annulus Reporting

In this section, we show that any data structure that solves 2D annulus reporting with $O(\log^{O(1)} n)$ query time must use $\mathring{\Omega}(n^3)$ space. Recall that a annulus is the region between two concentric circles and the *width* of the annulus is the difference between the radii of the two circles. In general, we show that if the query time is $Q(n) + O(k)$, then the data structure must use $\mathring{\Omega}(n^3/Q(n)^5)$ space. Note that this is also a better trade-off curve than what we obtained for the polynomial slab reporting problem when $\Delta = 2$. We will still use Chazelle's framework.

We first present a technical geometric lemma which upper bounds the intersection area of two 2D annuli. We will later use this lemma to show that with probability more than $1/2$, a random point set satisfies the first condition of Theorem 1.

▶ **Lemma 10.** *Consider two annuli of width w with inner radii of r_1, r_2, where $r_1 + w \leq r_2, w < r_1$, and $r_1, r_2 = \Theta(n)$. Let d be the distance between the centers of two annuli. When $w \leq d < r_2$, the intersection area of two annuli is bounded by $O\left(wn\sqrt{\frac{w^2}{(g+w)d}}\right)$, where $g = \max\{r_1 - r_2 + d, 0\}$.*

The proof sketch. For the complete proof see the full version of the paper. When $w \leq d \leq r_2 - r_1 + 2w$, the intersection region consists of two triangle-like regions. We only bound the triangle-like region $\tilde{\triangle}PQR$ in the upper half annuli as shown in Figure 1. We can show that its area is asymptotically upper bounded by the product of its base length $|QR| = O(w)$ and its height h. We bound h by observing that $\frac{hd}{2}$ is the area of triangle $\triangle PO_1O_2$ but we can also obtain its area of using Heron's formula, given its three side lengths. This gives $h = O(n\sqrt{w/d})$. Since in this case $g \leq 2w$, the intersection area is upper bounded by $O\left(wn\sqrt{\frac{w^2}{(g+w)d}}\right)$ as claimed.

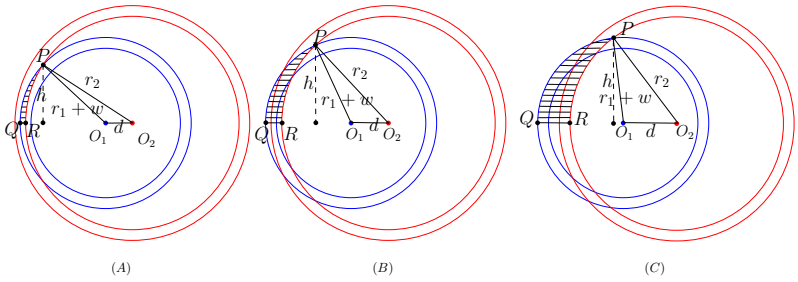
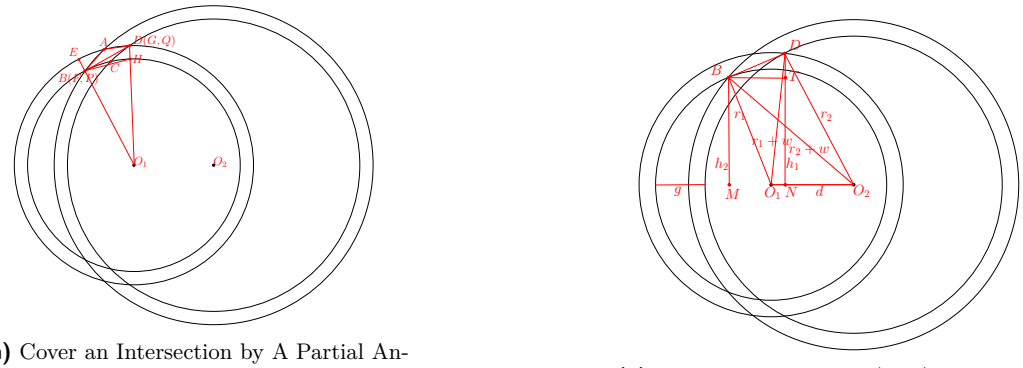


Figure 1 Intersections When d is Small.

When $r_2 - r_1 + 2w \leq d \leq r_2$, the intersection region consists of two quadrilateral-like regions. Again we only consider $\tilde{\square}ABCD$ in the upper half of the annuli, which is contained in a partial annulus, $\tilde{\mathcal{R}}_{EFHD}$, as shown in Figure 2a. We show the area of $\tilde{\mathcal{R}}_{EFHD}$ is asymptotically bounded by $|BH| \cdot w$, where $|BH|$ is the distance between the two endpoints of the inner arc. We upper bound $|BH|$ by $|BD|$. We use the algebraic representation of the two annuli, to bound the length of the projection of BD on the X-axis by $\Theta\left(\frac{wn}{d}\right)$; See Figure 2b. We use Heron's formula to bound the length of the projection of BD on the Y-axis by $O\left(n\sqrt{w^2/dg}\right)$. The maximum of the length of the two projections yields the claimed bound. ◀



(a) Cover an Intersection by A Partial Annulus

(b) Bound the Length of $|BD|$

Figure 2 Cover a Quadrilateral-like Region by a Partial Annulus.

We use Chazelle's framework to obtain a lower bound for 2D annulus reporting. Let S_1 and S_2 be two squares of side length n that are placed $10n$ distance apart and S_2 is directly to the left of S_1. We generate the annuli as follows. We divide S_1 into a $\frac{n}{T} \times \frac{n}{T}$ grid where each cell is a square of side length T. For each grid point, we construct a series of circles as follows. Let O be a grid point. The first circle generated for O must pass through a corner of S_2 and not intersect the right side of S_2, as shown in Figure 3. Then we create a series of circles centered at O by increasing the radius by increments of w, for some $w < T$, as long as it does not intersect the left side of S_2. Every consecutive two circles defines a annulus centered on O. We repeat this for every grid cell in S_1 and this makes up our set of queries. The input points are placed uniformly randomly inside S_2.

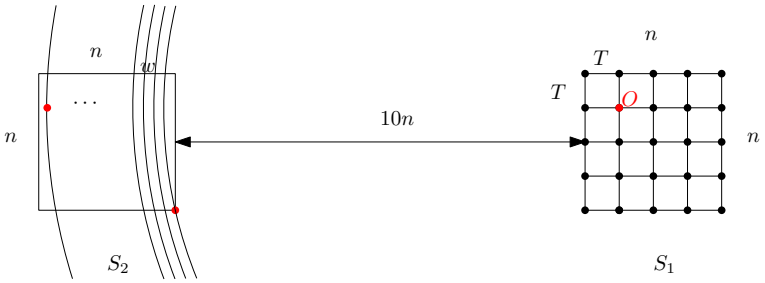


Figure 3 Generate a Family of Annuli at Point O.

We now show that for the annuli we constructed, the intersection of ℓ annuli is not too large, for some ℓ we specify later. More precisely we prove the following.

▶ **Lemma 11.** *There exists a large enough constant c such that in any subset of $\ell = cw^2/\sqrt{T}$ annuli, we can find two annuli such that their intersection has area $O\left(nw\sqrt{\frac{1}{T}}\right)$.*

The proof sketch. For the complete proof see the full version of the paper. Let $\mathcal{S}$ be a set of $\ell = cw^2/\sqrt{T}$ annuli. Suppose for the sake of contradiction that we cannot find two annuli in $\mathcal{S}$ whose intersection area is $O\left(nw\sqrt{\frac{1}{T}}\right)$. Since by Lemma 10, the intersection area of any two annuli in our construction with distance $\Omega(wT)$ is $O\left(nw\sqrt{\frac{1}{T}}\right)$. The maximum distance between any two annuli in $\mathcal{S}$ must be $o(wT)$.

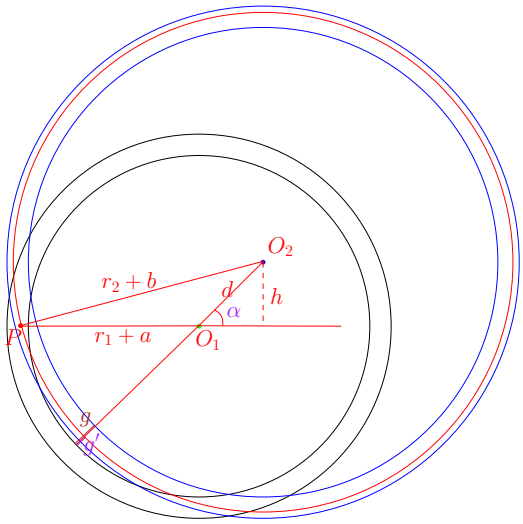


Figure 4 Intersection of Two Annuli.

Let P be a point in the intersection of annuli in $\mathcal{S}$. Consider an arbitrary annulus $\mathcal{R}_1 \in \mathcal{S}$ centered at O_1 and another annulus $\mathcal{R}_2 \in \mathcal{S}$ centered at O_2 for some $O_2 \notin PO_1$. For $\mathcal{R}_1, \mathcal{R}_2$ to contain P, we must have $|PO_1| = r_1 + a, |PO_2| = r_2 + b$ for $0 \leq a, b \leq w$. See Figure 4 for an example. Also $|O_1O_2| = d$, by exploiting the shape of $\triangle PO_1O_2$ and applying Lemma 10, we can compute an upper bound for the distance between O_2 and PO_1, namely, $h = d \sin \alpha = o(w\sqrt{T})$, where α is the angle between O_1O_2 and PO_1. This implies that $\mathcal{S}$ must fit in a rectangle of size $o(wT) \times o(w\sqrt{T})$. Since the gird cell size is $T \times T$, only $o(w^2/\sqrt{T})$ annuli are contained in such a rectangle, a contradiction. ◀

We are now ready to plug in some parameters in our construction. We set $T = w^2 2^{2\sqrt{\log n}}$. First, we claim that from each grid cell O, we can draw $\Theta(n/w)$ circles; Let C_1, C_2, C_3, and C_4 be the corners of S_2 sorted increasingly according to their distance to O. As S_1 and S_2 are placed $10n$ distance apart, an elementary geometric calculation reveals that C_1 and C_2 are vertices of the right edge of S_2, meaning, the smallest circle that we draw from O passes through C_2 and we keep drawing circles, by incrementing their radii by w until we are about to draw a circle that is about to contain C_3. We can see that $|OC_3| - |OC_2| = \Theta(n)$ and thus we draw $\Theta(n/w)$ circles from O. As we have $\Theta((n/T)^2)$ grid cells, it thus follows that we have $\Theta(n^3/(T^2w))$ annuli in our construction.

Also by our construction, the area of each annulus within S_2 is $\Theta(wn)$. To see this, let P be an arbitrary point in S_1, let A, B be the intersections of some circle centered at P as in Figure 5.

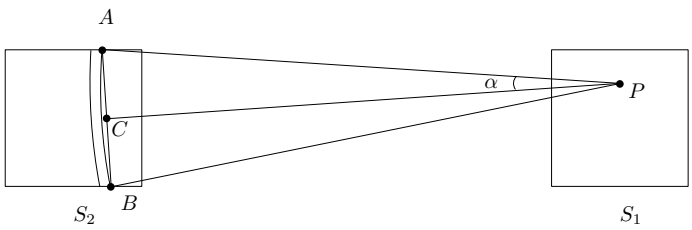


Figure 5 The Angle of a Annulus.

We connect AB and let C be the center of AB. Let $\alpha = \angle APC$. In the triangle $\triangle ABP$, all the sides are within constant factors of each other and thus $\alpha = \Theta(1)$ and so the area of the annulus inside S_2 is at least a constant fraction of the area of the entire annulus.

Suppose we have a data structure that answers 2D annulus reporting queries in $Q(n)+O(k)$ time. We set $w = c'Q(n)$ for a large enough constant c' such that the area of each annulus within S_2 is at least $\Theta(wn) > 8nQ(n)$. By Lemma 5, if we sample n points uniformly at random in S_2, then with probability more than $1/2$, each annulus contains at least $Q(n)$ points.

Also by our construction, the total number of intersections of two annuli is bounded by $O(n^6)$ and by our choice of T, $O\left(nw\sqrt{\frac{1}{T}}\right) = O\left(\frac{n}{2\sqrt{\log n}}\right)$. Then by Lemma 4 and Lemma 11, with probability $> \frac{1}{2}$, a point set of size n picked uniformly at random in S_2 satisfies that the number of points in any of the intersection of $cw^2/\sqrt{T}$ annuli is no more than $9\sqrt{\log n}$.

Now by union bound, there exist $\Theta\left(\frac{n^3}{wT^2}\right)$ point sets such that each set is the output of some 2D annulus query and each set contains at least $Q(n)$ points. Furthermore, the intersection of any $cw^2/\sqrt{T}$ sets is bounded by $9\sqrt{\log n}$. Then by Theorem 1, we obtain a lower bound of

$$S(n) = \Omega\left(\frac{Q(n)n^3\sqrt{T}}{wT^2w^2 2^{O\left(\sqrt{\log n}\right)}}\right) = \mathring{\Omega}\left(\frac{n^3}{Q(n)^5}\right).$$

This proves the following theorem about 2D annulus reporting.

▶ **Theorem 12.** *Any data structure that solves 2D annulus reporting on point set of size n with query time $Q(n) + O(k)$, where k is the output size, must use $\mathring{\Omega}\left(n^3/Q(n)^5\right)$ space.*

So for any data structure that solves 2D annulus reporting in time $Q(n) = O(\log^{O(1)} n)$, Theorem 12 implies that $\mathring{\Omega}\left(n^3\right)$ space must be used.

4.2 2D Annulus Stabbing

Modifications similar to those done in Subsection 3.2 can be used to obtain the following lower bound. See the full version of the paper for details.

▶ **Theorem 13.** *Any data structure that solves the 2D annulus stabbing problem with query time $Q(n) + O(k)$, where k is the output size, must use $S(n) = \Omega(n^{3/2}/Q(n)^{3/4})$ space.*

So for any data structure that solves the 2D annulus stabbing problem using $O(n)$ space, Theorem 13 implies that its query time must be $Q(n) = \Omega(n^{2/3})$.

5 Conclusion and Open Problems

We investigated lower bounds for range searching with polynomial slabs and annuli in $\mathbb{R}^2$. We showed space-time tradeoff bounds of $S(n) = \mathring{\Omega}(n^{\Delta+1}/Q(n)^{\Delta^2+\Delta})$ and $S(n) = \mathring{\Omega}(n^3/Q(n)^5)$ for them respectively. Both of these bounds are almost tight in the "fast query" case, i.e., when $Q(n) = O(\log^{O(1)} n)$ (up to a $n^{o(1)}$ factor). This refutes the conjecture of the existence of data structure that can solve semialgebraic range searching in $\mathbb{R}^d$ using $\mathring{O}(n^d)$ space and $O(\log^{O(1)} n)$ query time. We also studied the "dual" polynomial slab stabbing and annulus stabbing problems. For these two problems, we obtained lower bounds $S(n) = \Omega(n^{1+1/\Delta}/Q(n)^{(\Delta+1)/\Delta^2})$ and $S(n) = \Omega(n^{3/2}/Q(n)^{3/4})$ respectively. These bounds are tight when $S(n) = O(n)$. Our work, however, brings out some very interesting open problems.

To get the lower bounds for the polynomial slabs, we only considered univariate polynomials of degree Δ. In this setting, the number of coefficients is at most $\Delta + 1$, and we have also assumed they are all independent. It would be interesting to see if similar lower bounds can be obtained under more general settings. In particular, as the maximum number of coefficients of a bivaraite polynomial of degree Δ is $\binom{\Delta+2}{2}$, it would interesting to see if a $\mathring{\Omega}(n^{\binom{\Delta+2}{2}-1})$ space lower bound can be obtained for the "fast query" case.

It would also be interesting to consider space-time trade-offs. For instance, by combining the known "fast query" and "low space" solutions for 2D annulus reporting, one can obtain data structures with trade-off curve $S(n) = \tilde{O}(n^3/Q(n)^4)$, however, our lower bound is $S(n) = \mathring{\Omega}(n^3/Q(n)^5)$ and it is not clear which of these bounds is closer to truth. For the annulus searching problem in $\mathbb{R}^2$, in our lower bound proof, we considered a random input point set, since in most cases a random point set is the hardest input instance and our analysis seems to be tight, we therefore conjecture that our lower bound could be tight, at least when $Q(n)$ is small enough. We believe that it should be possible to obtain the trade-off curve of $S(n) = \tilde{O}(n^3/Q(n)^5)$ when the input points are uniformly random in the unit square and $Q(n)$ is not too big.

Finally, another interesting direction is to study the lower bound for the counting variant of semialgebraic range searching.

References

1. Peyman Afshani. Improved pointer machine and I/O lower bounds for simplex range reporting and related problems. In *Proceedings of the Twenty-Eighth Annual Symposium on Computational Geometry*, SoCG '12, page 339–346, New York, NY, USA, 2012. Association for Computing Machinery. doi:10.1145/2261250.2261301.
2. Peyman Afshani and Anne Driemel. On the complexity of range searching among curves. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 898–917. SIAM, Philadelphia, PA, 2018. doi:10.1137/1.9781611975031.58.
3. Pankaj K. Agarwal. *Simplex Range Searching and Its Variants: A Review*, pages 1–30. Springer International Publishing, Cham, 2017. doi:10.1007/978-3-319-44479-6_1.
4. Pankaj K. Agarwal, Boris Aronov, Esther Ezra, and Joshua Zahl. An efficient algorithm for generalized polynomial partitioning and its applications. In *35th International Symposium on Computational Geometry*, volume 129 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 5, 14. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2019.
5. Pankaj K. Agarwal and Jiří Matoušek. On range searching with semialgebraic sets. *Discrete Comput. Geom.*, 11(4):393–418, 1994. doi:10.1007/BF02574015.
6. Pankaj K. Agarwal, Jiří Matoušek, and Micha Sharir. On range searching with semialgebraic sets. II. *SIAM J. Comput.*, 42(6):2039–2062, 2013. doi:10.1137/120890855.
7. Timothy M. Chan. Optimal partition trees. *Discrete Comput. Geom.*, 47(4):661–690, 2012. doi:10.1007/s00454-012-9410-z.
8. B. Chazelle and J. Friedman. A deterministic view of random sampling and its use in geometry. *Combinatorica*, 10(3):229–249, 1990. doi:10.1007/BF02122778.
9. Bernard Chazelle. Lower bounds on the complexity of polytope range searching. *J. Amer. Math. Soc.*, 2(4):637–666, 1989. doi:10.2307/1990891.
10. Bernard Chazelle. Lower bounds for orthogonal range searching. I. The reporting case. *J. Assoc. Comput. Mach.*, 37(2):200–212, 1990. doi:10.1145/77600.77614.
11. Bernard Chazelle. Cutting hyperplanes for divide-and-conquer. *Discrete Comput. Geom.*, 9(2):145–158, 1993. doi:10.1007/BF02189314.
12. Bernard Chazelle. Cuttings. In Dinesh P. Mehta and Sartaj Sahni, editors, *Handbook of Data Structures and Applications*. Chapman and Hall/CRC, 2018. Second edition. doi:10.1201/9781315119335.

13 Bernard Chazelle and Burton Rosenberg. Simplex range reporting on a pointer machine. *Comput. Geom.*, 5(5):237–247, 1996. doi:10.1016/0925-7721(95)00002-X.

14 Bernard Chazelle and Emo Welzl. Quasi-optimal range searching in spaces of finite VC-dimension. *Discrete Comput. Geom.*, 4(5):467–489, 1989. doi:10.1007/BF02187743.

15 Kenneth L. Clarkson. New applications of random sampling in computational geometry. *Discrete Comput. Geom.*, 2(2):195–222, 1987. doi:10.1007/BF02187879.

16 Zeev Dvir. On the size of Kakeya sets in finite fields. *J. Amer. Math. Soc.*, 22(4):1093–1097, 2009. doi:10.1090/S0894-0347-08-00607-3.

17 Jacob E. Goodman, Joseph O'Rourke, and Csaba D. Tóth, editors. *Handbook of discrete and computational geometry*. Discrete Mathematics and its Applications (Boca Raton). CRC Press, Boca Raton, FL, 2018. Third edition. doi:10.1201/9781315119601.

18 Larry Guth and Nets Hawk Katz. On the Erdős distinct distances problem in the plane. *Ann. of Math. (2)*, 181(1):155–190, 2015. doi:10.4007/annals.2015.181.1.2.

19 D Haussler and E Welzl. Epsilon-nets and simplex range queries. In *Proceedings of the Second Annual Symposium on Computational Geometry*, SCG '86, page 61–71, New York, NY, USA, 1986. Association for Computing Machinery. doi:10.1145/10515.10522.

20 Jiří Matoušek. Cutting hyperplane arrangements. *Discrete Comput. Geom.*, 6(5):385–406, 1991. doi:10.1007/BF02574697.

21 Jiří Matoušek. Range searching with efficient hierarchical cuttings. *Discrete Comput. Geom.*, 10(2):157–182, 1993. doi:10.1007/BF02573972.

22 Jiří Matoušek and Zuzana Patáková. Multilevel polynomial partitions and simplified range searching. *Discrete Comput. Geom.*, 54(1):22–41, 2015. doi:10.1007/s00454-015-9701-2.

23 Emo Welzl. Partition trees for triangle counting and other range searching problems. In *Proceedings of the Fourth Annual Symposium on Computational Geometry (Urbana, IL, 1988)*, pages 23–33. ACM, New York, 1988. doi:10.1145/73393.73397.

24 Dan E. Willard. Polygon retrieval. *SIAM J. Comput.*, 11(1):149–165, 1982. doi:10.1137/0211012.

25 A C Yao and F F Yao. A general approach to d-dimensional geometric queries. In *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, STOC '85, page 163–168, New York, NY, USA, 1985. Association for Computing Machinery. doi:10.1145/22145.22163.

Rectilinear Steiner Trees in Narrow Strips

Henk Alkema
Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands

Mark de Berg
Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands

Abstract

A *rectilinear Steiner tree* for a set P of points in $\mathbb{R}^2$ is a tree that connects the points in P using horizontal and vertical line segments. The goal of Minimum Rectilinear Steiner Tree is to find a rectilinear Steiner tree with minimal total length. We investigate how the complexity of Minimum Rectilinear Steiner Tree for point sets P inside the strip $(-\infty, +\infty) \times [0, \delta]$ depends on the strip width δ. We obtain two main results.

- We present an algorithm with running time $n^{O(\sqrt{\delta})}$ for sparse point sets, that is, point sets where each $1 \times \delta$ rectangle inside the strip contains $O(1)$ points.
- For random point sets, where the points are chosen randomly inside a rectangle of height δ and expected width n, we present an algorithm that is fixed-parameter tractable with respect to δ and linear in n. It has an expected running time of $2^{O(\delta\sqrt{\delta})}n$.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases Computational geometry, fixed-parameter tractable algorithms

Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.9

Related Version *Full Version*: `https://arxiv.org/abs/2103.08354`

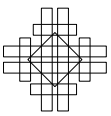
Funding The work in this paper is supported by the Netherlands Organisation for Scientific Research (NWO) through Gravitation-grant NETWORKS-024.002.003.

Acknowledgements We thank Remco van der Hofstad for discussions about the probabilistic analysis.

1 Introduction

In the Minimum Steiner Tree problem in the plane, we are given as input a set P of points in the plane, called *terminals*, and the goal is to find a minimum-length tree that connects the terminals in P. Thus the given terminals must be nodes of the tree, but the tree may also use so-called *Steiner points* as nodes. Minimum Steiner Tree is a classic optimization problem. It was among the first problems to be proven NP-hard, not only for the case where the length of the tree is measured using Euclidean metric [13] but also in the rectilinear version [14]. It was also shown to be NP-hard for other metrics [6]

The rectilinear version of the problem, where the edges of the tree must be horizontal or vertical, is one of the most widely studied variants, and it is also the topic of our paper. The Minimum Rectilinear Steiner Tree problem dates back more than 50 years [15, 16]. Its popularity arises from its many applications, in particular in the design of integrated circuits [7, 4, 5, 23]. The two most important early insights on Minimum Rectilinear Steiner Tree came from Hanan [16] and Hwang [17]. Hanan observed that any terminal set P admits a minimum rectilinear Steiner tree (MRST, for short) whose edges lie on the grid formed by all horizontal and vertical lines passing through at least one terminal in P. This grid is often called the *Hanan grid*. This implies that the Minimum Rectilinear Steiner Tree problem can be reduced to a purely combinatorial problem – namely, a Steiner-tree problem on graphs – which is not possible for the Euclidean version of the problem. Hwang investigated the structure of optimal MRSTs in more detail, by providing a characterization of the different components of an MRST; see Section 2.




As mentioned, Minimum Rectilinear Steiner Tree can be considered a special case of the Steiner-tree problem on graphs. Here the input is an edge-weighted graph $G = (V(G), E(G))$ and a terminal set $P \subseteq V(G)$, and the goal is to compute a minimum-length subtree of G that includes all terminals. In 1971 Dreyfus and Wagner [11] gave an algorithm solving the Steiner-Tree problem on graphs in time $3^n \cdot \log W \cdot |V(G)|^{O(1)}$, where W is the maximum edge weight in G. This was later improved by Björklund *et al.* [3] and Nederlof [19], who gave an algorithm with $2^n \cdot W \cdot |V(G)|^{O(1)}$ running time. A variant of the Dreyfus-Wagner algorithm for Minimum Rectilinear Steiner Tree runs in time $O(n^2 \cdot 3^n)$. Thobmorson *et al.* [21] and Deneen *et al.* [10] gave randomized algorithms for the special case of Minimum Rectilinear Steiner Tree where the terminals are drawn independently and uniformly from a rectangle. Both run in $2^{O(\sqrt{n}\log n)}$ expected time. Finally, in 2018 Fomin *et al.* [12] presented a $2^{O(\sqrt{n}\log n)}$ algorithm for general point sets.

Due to the many applications of Minimum Steiner Tree variants in the plane, there has also been significant interest in practical implementations. These implementations rely on the insight that a minimum Steiner tree can always be decomposed into so-called *full components*, which are maximal subtrees that do not have any terminals as internal nodes [17]. (This holds for the Euclidean as well as the rectilinear version.) To compute an exact solution, a set of candidate full components is first computed and then it is computed which subset of candidate full components can be concatenated into an MRST. This process was introduced by Winter in 1985 [24], in his software package *GeoSteiner*. Still, only very small data sets could be handled, and even in 1994 the state-of-the-art software could solve the rectilinear variant of the problem for only up to 16 points [20]. Warme's dissertation [22] significantly improved the process of concatenating the full components, resulting in optimal Steiner trees for up to 1,000 points for the rectilinear version of the problem and up to 2,000 points for the Euclidean version. In 1998 Althaus [2] obtained similar results. Throughout the years, GeoSteiner, which had become a collaboration between Warme, Winter and Zachariasen, has remained the fastest publicly available software package for computing minimum Steiner trees in the plane. By 2018, it could solve instances for up to 4,000 points for the rectilinear version, and up to 10,000 points for the Euclidean version [18].

Our contribution. The fastest known algorithm for Minimum Rectilinear Steiner Tree in $\mathbb{R}^2$ runs in $2^{O(\sqrt{n}\log n)}$ time [12]. In $\mathbb{R}$, on the other hand, the problem can be trivially solved in $O(n \log n)$ time by just sorting the points. In order to better understand the computational complexity of the classic Minimum Rectilinear Steiner Tree problem in the plane, we therefore investigate how the complexity depends on the width of the terminal set P. If the point set in P is "almost 1-dimensional" in the sense that the points lie in a narrow strip $\mathbb{R} \times [0, \delta]$, then can we solve Minimum Rectilinear Steiner Tree more efficiently than in the general case? And if so, how does the complexity scale with δ? Can we obtain an algorithm that is fixed-parameter tractable with respect to δ? This follows the line of research started recently by Alkema *et al.* [1], who studied these questions for the Traveling Salesman Problem. We study these questions in the following two scenarios.

- *Sparse point sets.* In this scenario, for any $x \in \mathbb{R}$ the rectangle $[x, x+1] \times [0, \delta]$ contains $O(1)$ points. We show that for sparse point sets in $\mathbb{R}^2$ an MRST must be k-tonic – an MRST is k-tonic if it intersects any vertical line at most k-times – for $k = O(\sqrt{\delta})$, and we give a dynamic-programming algorithm which runs in $n^{O(\sqrt{\delta})}$ time.
- *Random point sets.* Our main result is for point sets P generated randomly inside a rectangle of height δ and expected width n, as follows. First, we generate n independent exponentially distributed variables $\Delta_0, ..., \Delta_{n-1} \sim \text{Exp}(1)$. Using these, we compute the

x-coordinates of our points by setting x_i, the x-coordinate of the i-th point from P, as $x_i := \sum_{j=0}^{i-1} \Delta_i$ for $1 \leqslant i \leqslant n$. Next, we generate the y-coordinates of the points by picking each y_i uniformly and independently from the interval $[0, \delta]$. Thus the points from P lie inside the rectangle $[0, x_n] \times [0, \delta]$. One can show that asymptotically this distribution is essentially the same as the distribution obtained by picking n points uniformly at random from the rectangle $[0, n] \times [0, \delta]$ [9]. However, the random point process as just described is somewhat easier to analyze, so we will assume the points are generated according to that process. For this case we provide an FPT algorithm for Minimum Rectilinear Steiner Tree, which runs in expected time $2^{O(\delta\sqrt{\delta})}n$. More precisely, expected running time is $\min(n^{O(\sqrt{\delta})}, 2^{O(\delta\sqrt{\delta})}n)$. Note that the running time is linear when $\delta = O(1)$.

2 Preliminaries

Notation and terminology. Let $P := \{p_1, \ldots, p_n\}$ be a set of *terminals* in a 2-dimensional strip with height δ – we call such a strip a δ-*strip* – which we assume without loss of generality to be $\mathbb{R} \times [0, \delta]$. We use x_i and y_i to denote the x- and y-coordinate of point p_i, respectively. The points can be easily sorted on their x-coordinates: this can be done in $O(n \log n)$ time for sparse point sets, and in $O(n)$ expected time for random point sets [8]. Therefore, we will from now on assume that $x_i \leqslant x_j$ for all $1 \leqslant i \leqslant j \leqslant n$. We define the *spacing* of p_i (in P) as $\Delta_i := x_{i+1} - x_i$, for all $1 \leqslant i \leqslant n-1$. We write $P[i, j]$ to denote the set $\{p_i, ..., p_j\}$. We denote the vertical distance between two horizontal edges e, e' (or the horizontal distance between two vertical edges) by $\text{dist}(e, e')$.

Next we give some (mostly standard) terminology concerning rectilinear Steiner trees; see also Figure 1. A *rectilinear tree* is a tree structure embedded in the plane whose edges are horizontal or vertical line segments overlapping only at their endpoints. The *length* of a tree T, or $\|T\|$, is the sum of the lengths of its edges. A *rectilinear Steiner tree* for a set P of terminals is a rectilinear tree such that each terminal $p \in P$ is an endpoint of an edge in the tree. A *minimal rectilinear Steiner minimal tree (MRST)* is such a tree of minimum length.

The *degree* of a (Steiner or terminal) point q in a tree T is the number of edges incident on it. We denote the degree of q in T by $\text{degree}_T(q)$, or simply $\text{degree}(q)$ when T is clear from the context. Without loss of generality, if a degree-2 point has collinear (i.e. both horizontal or both vertical) incident edges then that point must be a terminal. Clearly, a point has degree at most 4. A point with degree of at least 3 that is not a terminal is called a *Steiner point*. A *corner* is a degree-2 point with non-collinear incident edges that is not a terminal. Hence, each endpoint of an edge is either a terminal, a Steiner point, or a corner.

A *segment* is defined to be a sequence of one or more adjacent collinear edges, with no terminals in the segments' interior.[1] A *complete segment* is a inclusion-wise maximal segment. Note that a complete segment does not have terminals in its interior. A corner is incident to exactly one horizontal complete segment and exactly one vertical complete segment. These complete segments are the *legs* of the corner. A *T-point* is a degree-3 Steiner point. Finally, a *cross* is a degree-4 Steiner point. Note that the endpoints of a complete segment are T-points, corners or terminals.

[1] When we refer to the "interior" of a segment, we always mean its relative interior, i.e. the segment excluding its endpoints.

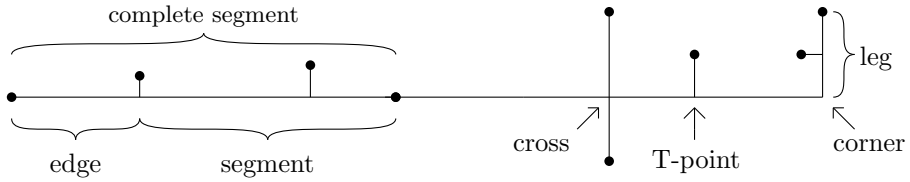


Figure 1 Illustration of terminology concerning rectilinear Steiner trees.

Separators will play a crucial role in our algorithms. A *separator* is a vertical line, not containing any of the points in P, that separates P into two non-empty subsets. For all $1 \leqslant i < n$ such that $x_i < x_{i+1}$, we define s_i to be the separator with x-coordinate $(x_i + x_{i+1})/2$. The *tonicity* of a rectilinear tree T *at a separator* s is the number of times T crosses s; when the tonicity of T at s is 1, we call it *monotonic* at s. The *tonicity* of a rectilinear tree T is the maximum over the tonicity of T at all separators. A rectilinear tree is called *monotonic* when its tonicity is 1.

Characterisation of the MRST. Over the years, many different properties of the MRST have been proven. One of the most important ones is the following:

▶ **Observation 1** (Hanan [16]). *Let P be a set of terminals in $\mathbb{R}^2$. Then there exists an MRST on P that is a subset of the* Hanan grid*, the grid formed by taking all horizontal and vertical lines which pass through at least one of the points of P.*

From now on, we will only consider rectilinear Steiner trees that lie on the Hanan grid. Furthermore, we can now directly conclude that the tonicity of an MRST is at most n.

A continuation on this characterisation is given by the Hwang theorem. We define a *full component* of a rectilinear Steiner tree T to be a maximal subtree that does not have any terminals as internal nodes. Note that a node in a full component of an MRST is a terminal if and only if it is a leaf in that component. Also note that any terminal $p_i \in P$ will be a leaf in exactly degree(p_i) full components. Hwang's theorem is now given by the following:

▶ **Theorem 2** (Hwang [17]). *Let P be a set of terminals in $\mathbb{R}^2$. Then there exists an MRST T on P with a maximal number of full components, such that each full component C is of one of the following four types. Let m_C be the number of terminals in C. Then C consists of*

- *four edges, connected in a cross,*
- *a single complete segment with $m_C - 2$ alternating incident edges,*
- *a corner and its legs, with $m_C - 2$ alternating edges incident to a single leg, or*
- *a corner and its legs, with $m_C - 3$ alternating edges incident to a single leg and a single edge incident to the other leg.*

For all legs, the incident edge closest to the corner must point away from the opposite leg. Furthermore, the edges incident to the long leg on the same side as the short leg are at least as long as the short leg.

We will call MRSTs which have this property *Hwang trees*. See Figure 2 for an example of each of the four types of full components of Hwang trees. Note that these full components do not contain a U-shape formed by an edge and two adjacent segments lying to the same side of that edge; any component with such a U-shape can be split into two full components by sliding the edge towards the terminals at the end of those segments. See Figure 3 for an example. We will call the complete segment with the $m_C - 2$ or $m_C - 3$ incident edges the *long leg*, and the other leg (if any) the *short leg*. If there are two complete segments which

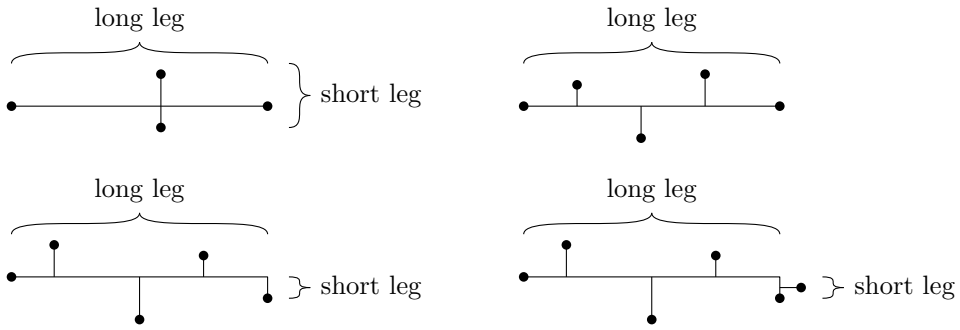


Figure 2 An example of each of the four different types of full components in Hwang trees.

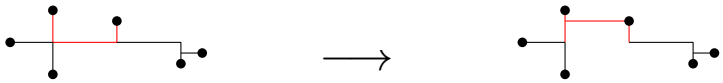


Figure 3 An example showing that Hwang trees do not contain a U-shape. The tree on the left has the same length as the tree on the right, but contains fewer full components. Therefore, the tree on the left is not a Hwang tree.

both have $m_C - 2$ or $m_C - 3$ incident edges, we will consider the horizontal one to be the long leg, and the vertical one to be the short leg. If the long leg is horizontal (vertical), we call the full component a *horizontal (vertical) full component*.

3 Sparse point sets inside a narrow strip

We say a point set is *sparse* if for all x the rectangle $[x, x+1] \times [0, \delta]$ contains at most k points for some arbitrary but fixed *sparseness constant* k. In this section, we will give a $n^{O(\sqrt{\delta})}$ algorithm for sparse point sets. We will do so in two steps. First, we will show that all separators are crossed at most $O(\sqrt{\delta})$ times. Then, we will give a dynamic-programming algorithm which sweeps from left to right and runs in the desired time.

First, we will show that parallel edges of an MRST cannot be too close. Recall that Δ_i denotes the horizontal spacing between p_i and p_{i+1}, and that δ denotes the height of the strip containing P. Also recall that s_i is the separator in between the points p_i and p_{i+1}.

▶ **Observation 3.**

(i) *Let $E = \{e_1, \ldots, e_m\}$ be a set of m horizontal edges of an MRST T which all intersect two vertical lines ℓ and ℓ'. Then $m \leqslant 1 + \lfloor \delta / \mathrm{dist}(\ell, \ell') \rfloor$. A similar statement holds when E is a set of vertical edges intersecting two horizontal lines.*

(ii) *If $\Delta_i > \delta$, then the tonicity of any MRST at s_i is 1.*

For the proof, see the full version. We are now ready to bound the tonicity at the separators. The following lemma will also be applicable for randomly generated point sets.

▶ **Lemma 4.** *Let T be a Hwang tree on P. Let s_i be a separator such that*

$$x_{i+\lceil\sqrt{\delta}\rceil+c_1} - x_i > c_2\sqrt{\delta}$$

for an integer constant $c_1 \geqslant 0$ and a constant $c_2 > 0$. Then the tonicity of T at s_i is $O(\sqrt{\delta}+1)$.

The proof of Lemma 4 can be found in the full version.

Using Lemma 4 we can now prove a bound on the tonicity of MRSTs of sparse point sets.

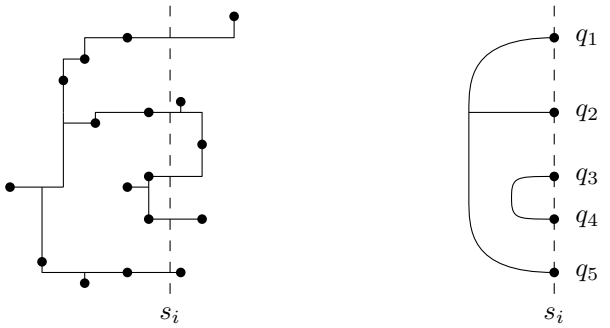


Figure 4 An example of an MRST and its crossing pattern $C = \{\{q_1, q_2, q_5\}, \{q_3, q_4\}\}$ at s_i.

▶ **Corollary 5.** *An MRST on a sparse point set P in a δ-strip is $\left((9k+18)(2+\sqrt{\delta})\right)$-tonic, where k is the sparseness constant.*

Proof. First, we note that since our point set P is sparse, we have $x_j - x_i \geqslant \lfloor (j-i)/k \rfloor$ for all $j > i$. Specifically, for all s_i such that $i + \lceil\sqrt{\delta}\rceil + k \leqslant n$, we get

$$x_{i+\lceil\sqrt{\delta}\rceil+k} - x_i \geqslant \left\lfloor \frac{\lceil\sqrt{\delta}\rceil + k}{k} \right\rfloor \geqslant \left\lfloor \frac{\sqrt{\delta}}{k} + 1 \right\rfloor \geqslant \frac{\sqrt{\delta}}{k}.$$

Therefore, we can invoke Lemma 4 with $c_1 = k$ and $c_2 = 1/k$, giving us that all these s_i are crossed at most $17k + 36 + (4k+17)\sqrt{\delta}$ times. (These constant follow from the constants in the proof of Lemma 4, see the Appendix.) By symmetry, we can do the same for all s_i such that $(i+1) - \lceil\sqrt{\delta}\rceil - k \geqslant 1$. Finally, we note that if this does not cover all s_i, then we have fewer than $17k + 36 + (4k+17)\sqrt{\delta}$ points in total. Since every separator is crossed at most n times, the statement also holds in this case. We conclude that for sparse terminal sets, all separators are crossed at most $17k + 36 + (4k+17)\sqrt{\delta} < (9k+18)(2+\sqrt{\delta})$ times. ◀

Corollary 5 gives rise to a natural dynamic-programming algorithm, as explained next. Let T be a rectilinear Steiner tree, and let s_i be a separator. We define the *crossing pattern* of T at s_i as follows. Let $X(s_i)$ be the set of at most n points where the Hanan grid crosses s_i, and let $X(s_i, T) \subseteq X(s_i)$ be the subset of points where T crosses s_i. If T is an MRST,

$$|X(s_i, T)| \leqslant (9k+18)(2+\sqrt{\delta}) = O(\sqrt{\delta})$$

by Corollary 5. We partition $X(s_i, T)$ into parts (that is, subsets) such that two points from $X(s_i, T)$ are in the same part if the path in T between these points fully lies to the left of s_i. The resulting partition of $X(s_i, T)$ is the crossing pattern of T at s_i; see Figure 4 for an example. We will say that a rectilinear forest T *adheres to* C at s_i if T lies fully to the left of s_i, and there exists a rectilinear forest T' which lies fully to the right of s_i such that $T \cup T'$ is a rectilinear Steiner tree with crossing pattern C at s_i. Note that not all crossing patterns can lead to an MRST: those that require crossing edges on the left-hand side (because they do not have a proper "nesting structure") can never lead to an MRST. We call the crossing patterns that contain at most $(9k+18)(2+\sqrt{\delta})$ points and do not require crossing edges on the left-hand side *viable crossing patterns*. We will now count the number of viable crossing patterns at s_i. There are $n^{O(\sqrt{\delta})}$ possible sets $X(s_i, T)$ that contain at most $(9k+18)(2+\sqrt{\delta})$ points. The number of viable partitions of these points – also known as the number of non-crossing partitions – follows the Catalan numbers. Hence, there are $2^{O(\sqrt{\delta})}$ possible viable partitions for each $X(s_i, T)$. This implies that the total number of viable crossing patterns for s_i is $n^{O(\sqrt{\delta})} \cdot 2^{O(\sqrt{\delta})} = n^{O(\sqrt{\delta})}$.

The algorithm. We can now define a table entry $A[i, X]$ for each separator s_i and viable crossing pattern X at s_i as follows.

$$A[i, X] := \text{the minimum length of a rectilinear forest adhering to } X \text{ at } s_i.$$

Note that the length of an MRST equals $A[n, \{\emptyset\}]$. Next we describe a recursive formula to compute the table entries. As a base case, we will use $A[0, X] = 0$ for $X = \{\emptyset\}$, and ∞ for all other X.

Let s_j and s_i be consecutive separators, with $j < i$. Note that since the point set is sparse, at most k points share an x-coordinate. Therefore, $j \geqslant i - k$. Let $F(X, s_i)$ be a minimum-length rectilinear forest adhering to X at s_i, and let X' be its (unknown) crossing pattern at s_j. Then the value of $A[i, X]$ equals the value of $A[j, X']$ plus the total length of the edges of $F(X, s_i)$ between s_j and s_i. The total length of $F(X, s_i)$ between these two separators only depends on X' and X. Since this subproblem contains $O(\sqrt{\delta})$ points with three different x-coordinates, its Hanan grid contains only $O(\sqrt{\delta})$ edges. Therefore, its value can be computed in $2^{O(\sqrt{\delta})}$ time by simply checking every possible subset of edges. Let $L(X', X)$ denote the total length of the solution to this subproblem. If no solution exists, we define it to be ∞. Then we get

$$A[i, X] = \min_{\text{viable } X'} A[j, X'] + L(X', X),$$

where s_j is the separator immediately preceding s_i, and the sum is over all crossing patterns X' that are viable at s_j.

The running time. To analyse the running time, we first determine the number of table entries. There are $O(n)$ separators, and we have already seen for every separator s_i there are $n^{O(\sqrt{\delta})}$ possible viable crossing patterns. Hence, the total number of table entries is $O(n) \cdot n^{O(\sqrt{\delta})} = n^{O(\sqrt{\delta})}$. Next, we calculate the time needed per table entry. For each of the $n^{O(\sqrt{\delta})}$ possible viable crossing patterns X' we compute $L(X', X)$ in $2^{O(\sqrt{\delta})}$ time. This brings the total time needed per table entry to $n^{O(\sqrt{\delta})}$.

Since we have $n^{O(\sqrt{\delta})}$ table entries, each needing $n^{O(\sqrt{\delta})}$ time, we conclude:

▶ **Theorem 6.** *Let P be a sparse point set of size n inside a δ-strip. Then we can compute an MRST on P in $n^{O(\sqrt{\delta})}$ time.*

4 Random point sets inside a narrow rectangle

In this section we give an algorithm with $\min\{n^{O(\sqrt{\delta})}, 2^{O(\delta\sqrt{\delta})}n\}$ expected running time for points generated randomly inside a rectangle of height δ and expected width n. Specifically, we assume the points in P are generated as follows. First, we generate n independent exponentially distributed variables $\Delta_0, ..., \Delta_{n-1} \sim \text{Exp}(1)$. Using these, we compute the x-coordinates of our points by setting $x_i := \sum_{j=0}^{i-1} \Delta_i$ for $1 \leqslant i \leqslant n$. Next, we generate the y-coordinates of the points by picking each y_i uniformly and independently from the interval $[0, \delta]$. Thus the points from P lie inside the rectangle $[0, x_n] \times [0, \delta]$. Since the spacings Δ_i are chosen from an exponential distribution of rate 1, we have $\mathbb{E}[x_n] = \mathbb{E}[\sum_{i=0}^{n-1} \Delta_i] = n$. (More precisely, $\sum_{i=0}^{n-1} \Delta_i$ converges to a normal distribution with mean n and variance $\sqrt{n}$.)

Recall that the algorithm for sparse point sets from the previous section, which had running time $n^{O(\sqrt{\delta})}$, was based on the fact that any separator s_i of a sparse point set is crossed only $O(\sqrt{\delta})$ times. Thus for each separator there are $n^{O(\sqrt{\delta})}$ different crossing patterns. Our main goal is now to change this algorithm into an algorithm for random point

sets that is fixed-parameter tractable with parameter δ. We face two difficulties. First, unlike in the case of sparse point sets, we cannot guarantee that all separators are crossed only $O(\sqrt{\delta})$ times. Second, even if a separator is crossed $O(\sqrt{\delta})$ times, the number of candidate crossing patterns can still be $n^{\Theta(\sqrt{\delta})}$, which is too much for an FPT algorithm. We overcome these difficulties as follows.

To deal with the first issue we will define a certain configuration of points and a corresponding separator – we will call such separator a *soft wall* – such that the separator is crossed only $O(\sqrt{\delta})$ times. Our new dynamic programming algorithm will have table entries for every soft wall instead of for every separator. We will prove that we expect to find sufficiently many soft walls, so that the expected number of points in between two consecutive soft walls only depends on δ (and not on n). This still leaves the second problem, because where a soft wall is crossed by an MRST may depend on points from P that are beyond the previous or next soft wall. Thus the number of crossing patterns can still be $n^{\Theta(\sqrt{\delta})}$. We therefore also devise a second type of wall, the *hard wall*. This is a vertical line ℓ through an input point p_i that will not be crossed at all by an edge of an MRST. The MRST will consist of two independent parts: an MRST for the points to the left of ℓ plus p_i itself, and an MRST for the points to the right of ℓ plus p_i itself. More generally, if we have a collection of hard walls then the subproblems between any two consecutive hard walls are completely independent. Hard walls will occur much less frequently than soft walls, but still the expected number of points in between two consecutive hard walls will be shown to depend only on δ. Hence, the number of crossing patterns we need to consider for the soft walls in between the two hard walls only depends on δ, giving us an FPT algorithm.

We first give pseudocode for the global algorithm. Recall that $P[i,j] := \{p_i, ..., p_j\}$. The constant 100 mentioned is not special; it is merely an arbitrary large enough constant.

Algorithm 1 ComputeMRST(P).

1: Compute a collection $\mathcal{W}_{\text{hard}} = \{\ell_0, \ldots, \ell_m\}$ of hard walls, as described below. The walls in $\mathcal{W}_{\text{hard}}$ are numbered from left to right, with ℓ_0 and ℓ_m being "hard walls" consisting of the leftmost and rightmost points of P, respectively.
2: **for** $i \leftarrow 0$ to $m-1$ **do**
3: Let p_j and $p_{j'}$ be the middle points of the hard walls ℓ_i and ℓ_{i+1}, respectively.
4: **if** $\delta < 100$ **then**
5: Compute an MRST T_i for $P[j,j']$ using the $2^{O(\sqrt{n}\log n)}$ algorithm by Fomin *et al.*
6: **else**
7: Compute a collection $\mathcal{W}_{\text{soft}} = \{t_1, \ldots, t_z\}$ of soft walls for $P[j,j']$, as described below.
8: Compute an MRST T_i for $P[j,j']$ using the dynamic-programming algorithm described in Section 3, but using the collection $\mathcal{W}_{\text{soft}}$ as separators (instead of using all separators between consecutive points), as described below.
9: **return** $T_0 \cup \cdots \cup T_{m-1}$.

Computing hard walls. Let $P[i, i+4]$ be a subset of points from P, and let ℓ be the vertical line through p_{i+2}. We call ℓ a *hard wall* if $P[i, i+4]$ has the following properties:

- $\Delta_j > \delta$ for all $i \leqslant j \leqslant i+3$
- $y_{i+1} < y_{i+2} < y_{i+3}$

See Figure 5 for an example of a hard wall. A hard wall indeed splits the problem into independent subproblems, as shown in the lemma below.

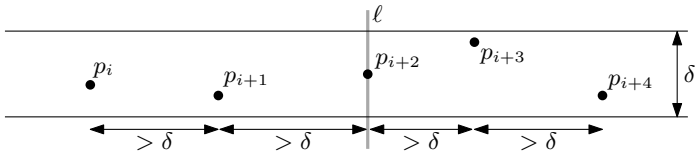


Figure 5 Example of a hard wall.

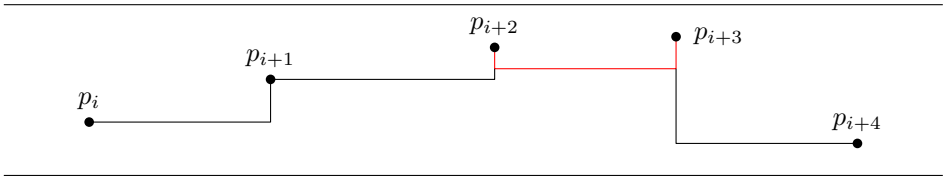


Figure 6 Illustration for the proof of Lemma 7 showing an MRST T where $\text{degree}_T(p_{i+2}) = 1$. In red, the U-shape showing that T is not a Hwang tree.

▶ **Lemma 7.** *Let ℓ be a hard wall, defined by the subset $P[i, i+4]$. Let T_1 be an MRST on $P[1, i+2]$ and let T_2 be an MRST on $P[i+2, n]$. Then $\|T\| = \|T_1\| + \|T_2\|$ and so $T_1 \cup T_2$ is an MRST on P.*

Proof. Let T be a Hwang tree on P. By Observation 3 we know that an MRST on P is monotonic at $s_i, ..., s_{i+3}$. The monotonicity at s_{i+1} and s_{i+2} implies that $\text{degree}_T(p_{i+2}) \leqslant 2$. If $\text{degree}_T(p_{i+2}) = 2$ then splitting T at p_{i+2} results in subtrees on $P[1, i+2]$ and $P[i+2, n]$ – this follows from the monotonicity at s_{i+1} and s_{i+2} – and so we are done. Now assume for a contradiction that $\text{degree}_T(p_{i+2}) = 1$. Then the incident edge if p_{i+2} must be vertical. Assume without loss of generality that p_{i+2} is the top endpoint of this edge. But then the (single) edge of T crossing s_{i+2} must reach the vertical line through p_{i+3} at a point q that lies somewhere below p_{i+3}. The monotonicity at s_{i+3} then implies that q must be connected to p_{i+3} by a vertical segment, thus creating a U-shape and contradicting that T is a Hwang tree. See Figure 6 for an example. ◀

The next lemma gives a bound on the probability that $P[i, i+4]$ is a hard wall.

▶ **Lemma 8.** $\mathbb{P}\Big[\, P[i, i+4] \text{ defines a hard wall} \,\Big] = e^{-4\delta}/6 \quad \text{for all } 1 \leqslant i \leqslant n-4.$

Proof. Recall that the spacings Δ_j are drawn from an exponential distribution with rate 1. Hence, $\mathbb{P}[\Delta_j] > \delta] = e^{-\delta}$ for all j. Since the spacings are independent, the probability that all four spacings between the points in $P[i, i+4]$ are greater than δ is $e^{-4\delta}$. Finally, $\mathbb{P}[y_{i+1} < y_{i+2} < y_{i+3}] = 1/6$, since all y-coordinates are chosen uniformly at random from $[0, \delta]$ and so all six orderings of $y_{i+1} < y_{y+2} < y_{i+3}$ are equally likely. ◀

The set $\mathcal{W}_{\text{hard}}$ of hard walls is now computed in the following straightforward manner: we check for all $i := 5j + 1$ with $j \in \{0, \ldots, \lfloor n/5 \rfloor - 1\}$ whether $P[i, i+4]$ defines a hard wall; if so, we add the corresponding hard wall to $\mathcal{W}_{\text{hard}}$. Note that this takes only $O(n)$ time in total, as each of the $O(n)$ candidate hard walls can be checked in $O(1)$ time.

Computing soft walls. Let $P[i, i + \lceil\sqrt{\delta}\rceil]$ be a subset of $\lceil\sqrt{\delta}\rceil + 1$ points from P such that $x_{i+\lceil\sqrt{\delta}\rceil} - x_i > \lceil\sqrt{\delta}\rceil/4$. Then we call the separator s_i – recall that s_i is the separator between p_i and p_{i+1} – a *soft wall*. See Figure 7 for an example.

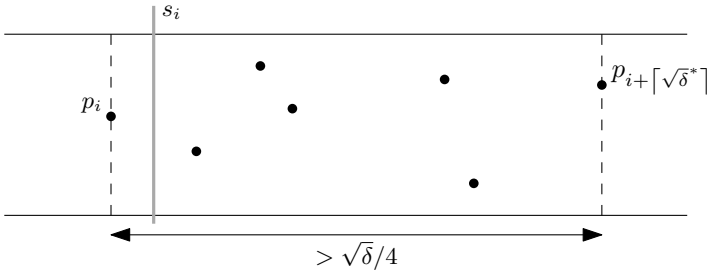


Figure 7 Example of a soft wall.

▶ **Lemma 9.** *Let $\delta \geqslant 100$. Let s_i be a soft wall, defined by $P[i, i+\lceil\sqrt{\delta}\rceil]$. Then s_i is crossed $O(\sqrt{\delta})$ times by an MRST. Furthermore, even under the assumption that $\Delta_j < \delta$ for all $1 \leqslant j \leqslant n-1$, we have*

$$\mathbb{P}\left[\, P\left[i, i+\left\lceil\sqrt{\delta}\right\rceil\right] \text{ defines a soft wall} \,\right] \;\geqslant\; 1 - 2^{3-\lceil\sqrt{\delta}\rceil/2} \text{ for all } 1 \leqslant i \leqslant n - \left\lceil\sqrt{\delta}\right\rceil.$$

Proof. The fact that s_i is crossed at most $O(1+\lceil\sqrt{\delta}\rceil) = O(\lceil\sqrt{\delta}\rceil)$ times follows immediately from Lemma 4. To be precise, s_i is crossed at most $18(2+\sqrt{\delta})$ times. It remains to derive a lower bound on the probability that $P[i, i+\lceil\sqrt{\delta}\rceil]$ is a soft wall, given that $\Delta_j < \delta$ for all $1 \leqslant j \leqslant n-1$. We have

$$\begin{aligned}
&\mathbb{P}\left[x_{i+\lceil\sqrt{\delta}\rceil} - x_i > \left\lceil\sqrt{\delta}\right\rceil/4\right] \\
&= 1 - \mathbb{P}\left[\sum_{j=i}^{i+\lceil\sqrt{\delta}\rceil-1} \Delta_j \leqslant \left\lceil\sqrt{\delta}\right\rceil/4\right] \\
&\geqslant 1 - \min_{t>0} e^{t\lceil\sqrt{\delta}\rceil/4} \left(\mathbb{E}[e^{-t\Delta_1}]\right)^{\lceil\sqrt{\delta}\rceil-1} && \text{by the Chernoff bound} \\
&\geqslant 1 - \min_{t>0} e^{t\lceil\sqrt{\delta}\rceil/4} \left(\int_{x=0}^{\delta} e^{-tx} e^{-x} \frac{1}{1-e^{-\delta}}\right)^{\lceil\sqrt{\delta}\rceil-1} && \Delta_1 \sim \text{Exp}(1) \text{ and } \Delta_1 \leqslant \delta
\end{aligned}$$

By taking $t = 3$, it can be shown that this is at least $1 - 2^{3-\lceil\sqrt{\delta}\rceil/2}$. ◀

Recall that in Algorithm ComputeMRST we need to compute soft walls for every subset $P[j, j']$ between two consecutive hard walls (including the points on those two hard walls). To this end we check whether $P[p_r, ..., p_{r+\lceil\sqrt{\delta}\rceil}]$ forms a soft wall for all $r := j + i\lceil\sqrt{\delta}\rceil$ with $0 \leqslant i \leqslant j' - \lceil\sqrt{\delta}\rceil$.

The dynamic-programming algorithm between two hard walls. Recall that $X(s_i)$ is the set of points where the Hanan grid crosses s_i. Let $\mathcal{X}_i$ denote the family of subsets of $X(s_i)$ of size at most $18(2+\sqrt{\delta})$. We can now define a table entry $A[i]$ for each *soft wall* s_i as follows.

$$A[i] := \text{a representative set of pairs } (X, l) \text{ where } l \text{ is the minimum length of a rectilinear forest adhering to } X \in \mathcal{X}_i \text{ at } s_i.$$

Here, "representative" means that for every soft wall s_i there exists an MRST T and $(X, l) \in A[i]$ such that T adheres to X at s_i. We will call this T an MRST *represented in* $A[i]$. See Figure 8 for an example. Note that $A[n]$ contains one element, of which l equals

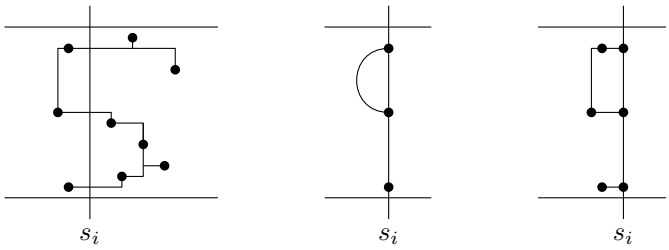


Figure 8 An example of an element of a table entry $A[i]$. On the left, an MRST T represented in $A[i]$. T is represented in $A[i]$ by a pair (X_i, l). In the middle, we have X_i. On the right, the edges contributing to the length l. Note that since T is an MRST, these edges indeed form a minimal length rectilinear forest adhering to X_i at s_i.

the length of an MRST on P. Next we describe a recursive formula to compute the table entries. As base case, we have $A[0] = \{\{\emptyset\}, 0\}$. We first give pseudocode for this part of the algorithm.

Algorithm 2 ComputeA(i).

```
1: Let s_j be the rightmost soft wall to the left of s_i.
2: for (X_j, l) ∈ A[j] do
3:     for all viable mirrored crossing patterns X'_i at s_i do
4:         Compute an MRST T' for the subproblem given by X_j and X_i.
5:         Add (X_i, l + ||T'||) to A[i], where X_i is the crossing pattern of T' at s_i.
6: Remove unviable pairs from A[i].
```

Let s_i be a soft wall, and let s_j be the rightmost soft wall to the left of s_i. We define a mirrored crossing pattern to be a crossing pattern where the partition denotes on how the rectilinear Steiner tree is connected on the *right* hand side. Let (X_j, l) be a pair in $A[j]$. Let T be an MRST adhering to X_j at s_j, and adhering to some (unknown) mirrored crossing pattern X'_i at s_i. Then there is a pair (X_i, l') in $A[i]$, where l' equals l plus the total length of the edges of T between s_j and s_i. The total length of T between these two separators only depends on X_j and X'_i. Let $L(X_j, X'_i)$ denote the total length of the solution to this subproblem. Now, to compute the value of $L(X_j, X'_i)$, we use the $2^{O(\sqrt{n}\log n)}$ algorithm by Fomin *et al.* [12]. Since this algorithm only computes Steiner trees (not forests adhering to some crossing pattern), we need to adapt our subproblem. To ensure the crossing pattern X_j, we mimic the edges on the left of X_j. For every part of X_j, we add a path of "virtual" edges of length 0, connecting the points in that part. These are automatically added to the so-called shortest path RST found by the first part of the algorithm by Fomin *et al.* Since the number of virtual edges added is constant in n, it does not affect its running time. We ensure the crossing pattern X'_i analogously. Given the output T' of the algorithm, we remove its virtual edges, and analyse its (non-mirrored) crossing pattern X_i at s_i. We then add the pair $(X_i, l + \|T'\|)$ to $A[i]$. After doing so for all pairs in $A[j]$ and viable mirrored crossing patterns X'_i, we may be able to remove some elements from $A[i]$. First, we remove any duplicates. Then, if two pairs have the same crossing pattern X_i, we need only the one with the smallest l.

We will now prove that $A[i]$ is indeed a representative set by induction on i. Clearly, $A[0]$ is a representative set. Now, suppose $A[j]$ is a representative set. We will now show that after performing the above, $A[i]$ is a representative set. See Figure 9 for an example. Since

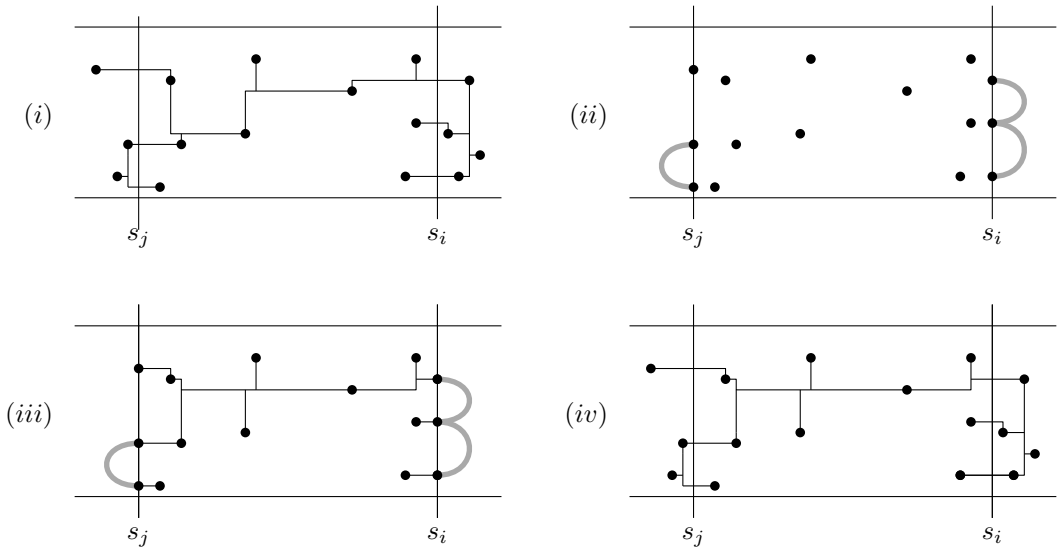


Figure 9 Illustration for the correctness proof of Algorithm 2. At (i), we have an MRST T which adheres to X_j at s_j, and to X'_i at s_i. At (ii), we have the corresponding subproblem. The thick grey edges denote the virtual edges of length 0. At (iii), a solution T' to the subproblem. Note that the total length of T' equals the total length of T between s_j and s_i. At (iv), the new MRST T'' represented in $A[i]$, obtained by combining T and T'.

$A[j]$ is a representative set, there exists a pair $(X_j, l) \in A[j]$ and an MRST T such that T adheres to X_j at s_j. Now, T adheres to some mirrored crossing pattern X'_i at s_i. Therefore, we will find an MRST T' on the subproblem defined by X_j and X'_i, and add a pair (X_i, l') to $A[i]$. Let T'' be the MRST on P obtained by exchanging the part of T between s_j and s_i for T'. Note that we can do that, since T and T' adhere to the same crossing patterns X_j and X'_i. Now, T'' is represented in $A[i]$ by (X_i, l').

Analysis of the running time. We now analyze the expected running time of Algorithm ComputeMRST. To do so, we will bound certain distributions by other distributions. To be precise, we bound the expected running time of any algorithm on a point set with a random number of points following a certain distribution by the expected running time of the algorithm on a point set with a differently distributed random number of points.

▶ **Observation 10** ([1]). *Let Y_1, Y_2 be two discrete nonnegative random variables, such that for all $k \geqslant 0$, the equation $\mathbb{P}[Y_1 \leqslant k] \geqslant \mathbb{P}[Y_2 \leqslant k]$ holds. Let $f(k)$ be an increasing nonnegative function such that $\mathbb{E}[f(Y_2)] < \infty$. Then*

$$\mathbb{E}[f(Y_1)] = \sum_{k=0}^{\infty} f(k)\mathbb{P}[Y_1 = k] \leqslant \sum_{k=0}^{\infty} f(k)\mathbb{P}[Y_2 = k] = \mathbb{E}[f(Y_2)].$$

We write $Y_1 \preccurlyeq Y_2$ to denote that for all $k \geqslant 0$, the equation $\mathbb{P}[Y_1 \leqslant k] \geqslant \mathbb{P}[Y_2 \leqslant k]$ holds.

Let us take a look at the sizes of the subproblems defined by the hard walls. Suppose we are computing $\mathcal{W}_{hard}$ and have just found a hard wall ℓ_i. Let the random variable X_1 denote the number of points in the subproblem $P[j, j']$ between the two hard walls ℓ_i and the unknown ℓ_{i+1}. Note that X_1 is at most $m := n - j + 1$, and that X_1 only depends on m. Therefore, we will write $X_{1,m}$. Now, $X_{1,m}$ is almost geometrically distributed. There are two differences: we only check whether $P[i, i + 4]$ defines a hard wall for i of the form $5j + 1$, and

$X_{1,m}$ is at most m. Since the probability that $P[i, i+4]$ defines a hard wall is $e^{-4\delta}/6$, we have $X_{1,m} \sim \min\{m, 1 + 5 \cdot \text{Geom}(e^{-4\delta}/6)\}$. Here, the probability mass function of $\text{Geom}(p)$ is $(1-p)^{k-1}p$. Let X_2 be the same distribution, but where we ignore the maximum number of points, $X_2 \sim 1 + 5 \cdot \text{Geom}(e^{-4\delta}/6)$. Then, $X_{1,m} \preccurlyeq X_2$ for all m.

We are now ready to calculate the expected running time of ComputeMRST if $\delta < 100$. We have already seen that we can find $\mathcal{W}_{\text{hard}}$ in $O(n)$ time. Since $X_{1,m} \preccurlyeq X_2$ for all m, the expected time needed per subproblem is bounded by the expected time needed to run the $2^{O(\sqrt{n}\log n)}$ algorithm by Fomin *et al.* on a point set with X_2 points. We get:

$$\sum_{k=1}^{\infty}(1-e^{-4\delta}/6)^{k-1} \cdot \left(e^{-4\delta}/6\right) \cdot 2^{O(\sqrt{5k+1}\log(5k+1))} < \sum_{k=1}^{\infty} 2^{-\Theta(k)} \cdot 2^{O(\sqrt{5k}\log(5k))} = O(1)$$

Since there are $O(n)$ subproblems, this finishes the case $\delta < 100$.

We can use the same trick for the distribution of the number of points between soft walls. Here, we let the random variable $Y_{1,m}$ denote the number of points between two consecutive soft walls, given that we have found no hard walls between the hard walls defining our subproblem and where m is once more the maximum number of points. We can bound $Y_{1,m}$ in three steps. First, note that the condition that no Δ_j is larger than δ is stronger than the condition that there are no hard walls between the hard walls defining our subproblem. Let $Y_{2,m}$ denote the number of points between the soft walls, given that no Δ_j is larger than δ. Then $Y_{1,m} \preccurlyeq Y_{2,m}$. Next, recall that if $\delta \geqslant 100$, by Lemma 9 the probability that s_i is a soft wall is at least $1 - 2^{3-\lceil\sqrt{\delta}\rceil/2}$, even if all $\Delta_j < \delta$. Define $Y_{3,m} \sim \min\{m, \lceil\sqrt{\delta}\rceil \cdot \text{Geom}(1 - 2^{3-\lceil\sqrt{\delta}\rceil/2})\}$. Then $Y_{2,m} \preccurlyeq Y_{3,m}$. Finally, analogously to the hard walls, we can remove the maximum number of points. Define $Y_4 \sim \lceil\sqrt{\delta}\rceil \cdot \text{Geom}(1 - 2^{3-\lceil\sqrt{\delta}\rceil/2})$. We conclude that $Y_{1,m} \preccurlyeq Y_4$ for all m.

Let $c, \lambda > 0$ be such that the algorithm by Fomin *et al.* runs in under $c \cdot 2^{\lambda\sqrt{n}\log n}$ time. For the case $\delta \geqslant 100$, the total expected time needed per subsubproblem is then bounded by

$$\begin{aligned}
&\sum_{i=1}^{\infty} c \cdot 2^{\lambda\sqrt{i\lceil\sqrt{\delta}\rceil+1}\log\left(i\lceil\sqrt{\delta}\rceil+1\right)} \cdot \left(2^{3-\lceil\sqrt{\delta}\rceil/2}\right)^{i-1} \cdot \left(1 - 2^{3-\lceil\sqrt{\delta}\rceil/2}\right) \\
&< c \cdot 2^{\lceil\sqrt{\delta}\rceil/2-3} \sum_{i=1}^{\infty} 2^{\lambda\sqrt{2i\lceil\sqrt{\delta}\rceil}\log\left(2i\lceil\sqrt{\delta}\rceil\right)} \cdot 2^{(3-\lceil\sqrt{\delta}\rceil/2)i} \cdot 1 \\
&< c \cdot 2^{\lceil\sqrt{\delta}\rceil} \sum_{i=1}^{\infty} 2^{4\lambda(2i\lceil\sqrt{\delta}\rceil)^{3/4}} \cdot 2^{-i\lceil\sqrt{\delta}\rceil/5} \qquad \text{since } \delta \geqslant 100 \\
&< c \cdot 2^{\lceil\sqrt{\delta}\rceil} \sum_{i=1}^{\infty} 2^{\lceil\sqrt{\delta}\rceil(8\lambda i^{3/4} - i/5)}
\end{aligned}$$

Now, for a sufficiently large M, we have $8\lambda i^{3/4} - i/5 < -i/10$ for all $i \geqslant M$. We get:

$$\begin{aligned}
&c \cdot 2^{\lceil\sqrt{\delta}\rceil} \sum_{i=1}^{\infty} 2^{\lceil\sqrt{\delta}\rceil(8\lambda i^{3/4} - i/5)} \\
&= c \cdot 2^{\lceil\sqrt{\delta}\rceil} \left(\sum_{i=1}^{M} 2^{\lceil\sqrt{\delta}\rceil(8\lambda i^{3/4} - i/5)} + \sum_{i=M+1}^{\infty} 2^{\lceil\sqrt{\delta}\rceil(8\lambda i^{3/4} - i/5)} \right) \\
&< c \cdot 2^{\lceil\sqrt{\delta}\rceil} \left(M \cdot 2^{\lceil\sqrt{\delta}\rceil \max_{i\geqslant 1}(8\lambda i^{3/4} - i/5)} + \sum_{i=M+1}^{\infty} 2^{-i/10} \right) = 2^{O(\lceil\sqrt{\delta}\rceil)}.
\end{aligned}$$

Let m be the number of points in the corresponding subproblem defined by two hard walls. Note that the above bound is independent of m. Analogously to the original sparse point-set algorithm, there are $m^{O(\lceil\sqrt{\delta}\rceil)}$ possible crossing patterns per separator. In total, this algorithm therefore takes $m \cdot m^{O(\lceil\sqrt{\delta}\rceil)} \cdot 2^{O(\lceil\sqrt{\delta}\rceil)} = m^{O(\sqrt{\delta})}$ expected time.

Now, all that remains is calculating the expected running time of our main random point set algorithm in this case. Clearly, it runs in at most expected $n^{O(\sqrt{\delta})}$ time, since splitting up the problem using the hard walls can only speed up the algorithm.

Let $\lambda, \mu \geqslant 1$ be such that the $m^{O(\sqrt{\delta})}$ expected running time algorithm runs in at most $m^{\lambda\sqrt{\delta}}$ expected time, and that the probability of a hard wall is $2^{-\mu\delta}$. Let Y_1 be the distribution of the number of points of a subproblem. Recall that $Y_1 \preccurlyeq 1 + 5 \cdot \mathrm{Geom}(e^{-4\delta}/6)$. Then the total expected time needed per subproblem is bounded by

$$\mathbb{E}\left[Y_1^{\lambda\sqrt{\delta}}\right] \leqslant \sum_{k=1}^{\infty}(1-2^{-\mu\delta})^{k-1} \cdot 2^{-\mu\delta} \cdot (5k+1)^{\lambda\sqrt{\delta}} < O(1) + 2^{-\mu\delta}\sum_{k=2}^{\infty} 2^{4\lambda\sqrt{\delta}\log k - 2^{-\mu\delta}k}.$$

Splitting this sum as well (see the full version for details), it can be shown that:

$$2^{-\mu\delta}\sum_{k=2}^{\infty} 2^{4\lambda\sqrt{\delta}\log k - 2^{-\mu\delta}k} < 2^{O(\delta\sqrt{\delta})}.$$

This brings the total expected running time to $O(n) \cdot (O(1) + 2^{O(\delta\sqrt{\delta})}) = 2^{O(\delta\sqrt{\delta})}n$.

All in all, our main random point set algorithm run in $O(n)$ expected time if $\delta < 100$, and in $\min\{n^{O(\sqrt{\delta})}, 2^{O(\delta\sqrt{\delta})}n\}$ expected time if $\delta \geqslant 100$. We conclude:

▶ **Theorem 11.** *Let P be a set of n points generated randomly inside a rectangle of height δ and expected width n, generated according to the procedure described earlier. Then an MRST on P can be found in $\min\{n^{O(\sqrt{\delta})}, 2^{O(\delta\sqrt{\delta})}n\}$ expected time.*

5 Concluding remarks

Our paper contains two main results on Minimum Rectilinear Steiner Tree. First, we proved that for sparse point sets in a strip of width δ, an MRST can be found in $n^{O(\sqrt{\delta})}$ time. Second, we gave a $\min\{n^{O(\sqrt{\delta})}, 2^{O(\delta\sqrt{\delta})}n\}$ expected running time algorithm for random point sets. For $\delta = \Theta(n)$ the running time equals the $2^{O(\sqrt{n}\log n)}$ of the algorithm for arbitrary point sets in the plane [12]. A challenging open problem is to see if an algorithm with running time $2^{O(\sqrt{\delta}\log\delta)}\mathrm{poly}(n)$ is possible. Another direction for future research is to study the problem in higher dimensions. We believe that our algorithmic results may carry over to $\mathbb{R}^d$ to points that are almost collinear, that is, that lie in a narrow cylinder. Generalizing the results to, say, points lying in a narrow slab will most likely be more challenging.

More generally, we believe that it is interesting to study the parameterized complexity of geometric problems using a "geometric parameter". For problems involving planar point sets, the strip width δ is a natural parameter, which is interesting because it explores the boundary between the 1-dimensional and 2-dimensional version of the problem. We have studied this for TSP in a previous paper [1] and for Minimum Rectilinear Steiner Tree in the current paper, but many other problems can be studied from this perspective as well.

References

1. Henk Alkema, Mark de Berg, and Sándor Kisfaludi-Bak. Euclidean TSP in narrow strips. In *Proc. 36th International Symposium on Computational Geometry (SoCG 2020)*, volume 164 of *LIPIcs*, pages 4:1–4:16, 2020. doi:10.4230/LIPIcs.SoCG.2020.4.
2. E. Althaus. Berechnung optimaler Steinerbäume in der ebene. Master's thesis, Max-Planck-Institutfür Informatik in Saarbrücken, Universität des Saarlandes, 1998.
3. Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Fourier meets Möbius: fast subset convolution. In *Proc. 39th Annual ACM Symposium on Theory of Computing (STOC 2007)*, pages 67–74. ACM, 2007. doi:10.1145/1250790.1250801.
4. Marcus Brazil, Doreen A. Thomas, Jia F. Weng, and Martin Zachariasen. Canonical forms and algorithms for Steiner trees in uniform orientation metrics. *Algorithmica*, 44(4):281–300, 2006. doi:10.1007/s00453-005-1178-6.
5. Marcus Brazil and Martin Zachariasen. Steiner trees for fixed orientation metrics. *J. Glob. Optim.*, 43(1):141–169, 2009. doi:10.1007/s10898-008-9305-y.
6. Marcus Brazil and Martin Zachariasen. The uniform orientation Steiner tree problem is NP-hard. *Int. J. Comput. Geom. Appl.*, 24(2):87–106, 2014. doi:10.1142/S0218195914500046.
7. Marcus Brazil and Martin Zachariasen. *Optimal Interconnection Trees in the Plane*, volume 29. Springer, May 2015. doi:10.1007/978-3-319-13915-9.
8. T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms (3rd edition)*. MIT Press, 2009.
9. D.J. Daley and D. Vere-Jones. *An Introduction to the Theory of Point Processes: Volume II: General Theory and Structure*. Probability and Its Applications. Springer New York, 2007. URL: https://books.google.nl/books?id=nPENXKw5kwcC.
10. Linda Deneen, Gary Shute, and Clark Thomborson. A probably fast, provably optimal algorithm for rectilinear Steiner trees. *Random Structures & Algorithms*, 5:535–557, October 1994. doi:10.1002/rsa.3240050405.
11. S. E. Dreyfus and R. A. Wagner. The steiner problem in graphs. *Networks*, 1(3):195–207, 1971. doi:10.1002/net.3230010302.
12. Fedor Fomin, Daniel Lokshtanov, Sudeshna Kolay, Fahad Panolan, and Saket Saurabh. Subexponential algorithms for rectilinear Steiner tree and arborescence problems. *ACM Transactions on Algorithms*, 16:1–37, March 2020. doi:10.1145/3381420.
13. M. R. Garey, R. L. Graham, and D. S. Johnson. The complexity of computing Steiner minimal trees. *SIAM Journal on Applied Mathematics*, 32(4):835–859, 1977. URL: http://www.jstor.org/stable/2100193.
14. M. R. Garey and D. S. Johnson. The rectilinear Steiner tree problem is NP-complete. *SIAM Journal on Applied Mathematics*, 32(4):826–834, 1977. URL: http://www.jstor.org/stable/2100192.
15. E. N. Gilbert and H. O. Pollak. Steiner minimal trees. *SIAM Journal on Applied Mathematics*, 16(1):1–29, 1968. URL: http://www.jstor.org/stable/2099400.
16. M. Hanan. On Steiner's problem with rectilinear distance. *SIAM Journal on Applied Mathematics*, 14(2):255–265, 1966. URL: http://www.jstor.org/stable/2946265.
17. F. K. Hwang. On Steiner minimal trees with rectilinear distance. *SIAM Journal on Applied Mathematics*, 30(1):104–114, 1976. URL: http://www.jstor.org/stable/2100587.
18. Daniel Juhl, David Warme, Pawel Winter, and Martin Zachariasen. The geoSteiner software package for computing Steiner trees in the plane: an updated computational study. *Mathematical Programming Computation*, 10:487–532, 2018. doi:10.1007/s12532-018-0135-8.
19. Jesper Nederlof. Fast polynomial-space algorithms using inclusion-exclusion. *Algorithmica*, 65(4):868–884, 2013. doi:10.1007/s00453-012-9630-x.
20. Clark D. Thomborson, Bowen Alpern, and Larry Carter. Rectilinear Steiner tree minimization on a workstation. In *Proce. DIMACS Workshop on Computational Support for Discrete Mathematics*, volume 15 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 119–136, 1992. doi:10.1090/dimacs/015/10.

21 Clark D. Thomborson, Linda L. Deneen, and Gary M. Shute. Computing a rectilinear Steiner minimal tree in $n^{O(\sqrt{n})}$ time. In *Proc. International Workshop on Parallel Algorithms and Architectures*, volume 269 of *Lecture Notes in Computer Science*, pages 176–183, 1987. doi:10.1007/3-540-18099-0_44.

22 David Warme. *Spanning Trees in Hypergraphs with Applications to Steiner Trees*. PhD thesis, University of Virginia, 1998.

23 Peter Widmayer, Ying-Fung Wu, and C. K. Wong. On some distance problems in fixed orientations. *SIAM J. Comput.*, 16(4):728–746, 1987. doi:10.1137/0216049.

24 Pawel Winter. An algorithm for the steiner problem in the euclidean plane. *Networks*, 15(3):323–345, 1985. doi:10.1002/net.3230150305.

Characterizing Universal Reconfigurability of Modular Pivoting Robots

Hugo A. Akitaya
University of Massachusetts Lowell, MA, USA

Erik D. Demaine
Massachusetts Institute of Technology, Cambridge, MA, USA

Andrei Gonczi
Tufts University, Medford, MA, USA

Dylan H. Hendrickson
Massachusetts Institute of Technology, Cambridge, MA, USA

Adam Hesterberg
Harvard University, Cambridge, MA, USA

Matias Korman
Siemens Electronic Design Automation, Portland, OR, USA

Oliver Korten
Columbia University, New York, NY, USA

Jayson Lynch
University of Waterloo, ON, Canada

Irene Parada
TU Eindhoven, The Netherlands

Vera Sacristán
Universitat Politècnica de Catalunya, Barcelona, Spain

Abstract

We give both efficient algorithms and hardness results for reconfiguring between two connected configurations of modules in the hexagonal grid. The reconfiguration moves that we consider are "pivots", where a hexagonal module rotates around a vertex shared with another module. Following prior work on modular robots, we define two natural sets of hexagon pivoting moves of increasing power: restricted and monkey moves. When we allow both moves, we present the first universal reconfiguration algorithm, which transforms between any two connected configurations using $O(n^3)$ monkey moves. This result strongly contrasts the analogous problem for squares, where there are rigid examples that do not have a single pivoting move preserving connectivity. On the other hand, if we only allow restricted moves, we prove that the reconfiguration problem becomes PSPACE-complete. Moreover, we show that, in contrast to hexagons, the reconfiguration problem for pivoting squares is PSPACE-complete regardless of the set of pivoting moves allowed. In the process, we strengthen the reduction framework of Demaine et al. [FUN'18] that we consider of independent interest.

2012 ACM Subject Classification Theory of computation → Computational geometry

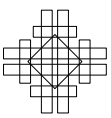
Keywords and phrases reconfiguration, geometric algorithm, PSPACE-hardness, pivoting hexagons, pivoting squares, modular robots

Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.10

Related Version Due to lack of space a large number of proofs are omitted.
Full Version: `https://arxiv.org/abs/2012.07556` [2]

Funding *Jayson Lynch*: Supported by NSERC.
Vera Sacristán: Partially supported by MTM2015-63791-R (MINECO/FEDER) and Gen. Cat. DGR 2017SGR1640.

Acknowledgements This research started at the 34th Bellairs Winter Workshop on Computational Geometry in 2019. We want to thank all participants for the fruitful discussions and a stimulating environment. We would also like to thank a SoCG reviewer for their many contributions that helped improve the presentation of the paper.



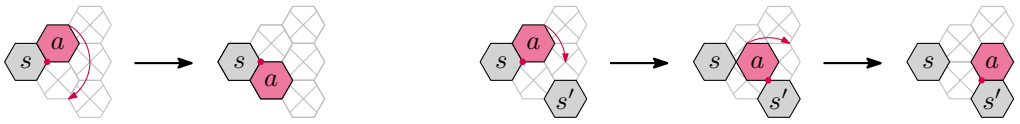
37th International Symposium on Computational Geometry (SoCG 2021).
Editors: Kevin Buchin and Éric Colin de Verdière; Article No. 10; pp. 10:1–10:20

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Reconfiguration problems encompass a large family of problems in which we need to provide a sequence of steps to transform one object into another. In this paper we consider the problem of reconfiguring a collection of modular robots (referred in this paper as **modules**) in a lattice using some prespecified set moves. Many variants of this problem have been studied both in the robotics and in the computational geometry communities. In this paper we study the reconfiguration problem for edge-connected configurations of hexagonal and of square modules. We follow the commonly used **single backbone condition** [11], that requires edge-connectivity to be maintained at all times. The moves allowed are pivots: a module can rotate around vertices shared with other modules and at the end of a move the pivoting module must lie in a lattice cell. The interior of two modules can never intersect.

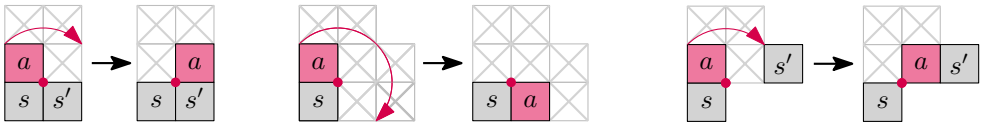
A hexagonal module can perform only two types of pivoting moves, illustrated in Figure 1. In a **restricted** move a module a adjacent to a module s pivots around a vertex v shared by a and s and ends the pivoting move in the other cell that has v on the boundary. The **restricted model** of pivoting only allows this move. In a **monkey** move a module a adjacent to a module s starts pivoting around a vertex v shared by a and s as in the restricted move, but halfway through the rotation another vertex w of a coincides with the vertex of a module s'. Then a continues the move pivoting around w in the same direction (clockwise or counterclockwise) as before until reaching a cell adjacent to s'. The **monkey model** of pivoting includes both the restricted and the monkey moves. Informally, the monkey move allows a module to keep pivoting in the same direction when a restricted move is not possible.



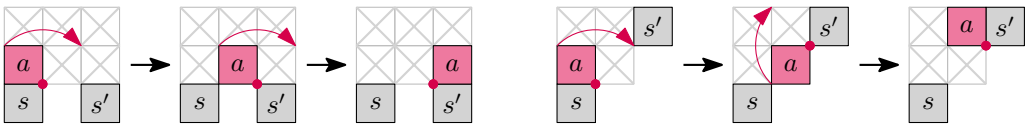
(a) Restricted move. **(b)** Monkey move.

Figure 1 Pivoting moves for hexagonal modules and their free-space requirements.

In the square grid two modules that share a vertex might not share an edge. Thus, for square modules there is a greater variety of pivoting moves. The three different sets of moves are illustrated in Figure 2. The **restricted model** includes only restricted moves, the **leapfrog model** includes both restricted and leapfrog moves, and the **monkey model** includes all moves.



(a) Restricted moves. **(b)** Leapfrog move.



(c) Monkey moves.

Figure 2 Pivoting moves for square modules and their free-space requirements.

Related work and contribution

One of the most natural questions for modular robots is whether universal reconfiguration is possible. That is, is there an algorithm to transform any (connected) configuration of n modules into another configuration with the same number of modules?

Efficient algorithms are known for universal reconfiguration of modular robots using moves that have significantly lighter free-space requirements [3, 10, 11, 12]. Relaxing the connectivity requirement has also lead to reconfigurability results [7].

The setting of this paper (pivoting robots) has proven to be more challenging. Instead, previous work has revolved around providing sufficient conditions for reconfiguration. Nguyen, Guibas and Kim [15] showed that reconfiguration of hexagonal modules using only restricted moves is always possible between configurations without the forbidden pattern illustrated in Figure 3 (left). Similarly, for pivoting squares, Sung et al. [16] presented an algorithm for reconfiguring between configurations without the patterns shown in Figure 3 (right). These algorithms do not provide reconfiguration guarantees as soon as the configuration contains a single copy of the forbidden pattern. In an attempt to remove global requirements, a recent result [1] introduced a different type of necessary condition: an efficient algorithm for reconfiguring between any two configurations that have 5 modules on the external boundary that can freely move (for pivoting squares in the monkey model). Other algorithms to reconfigure pivoting squares and hexagons are heuristics that do not provide termination guarantees [5, 14].

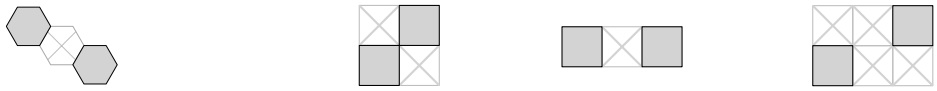


Figure 3 Forbidden patterns in previous algorithms for hexagonal and square pivoting modules.

Despite many attempts, universal reconfiguration remains unsolved in the setting of edge-connected pivoting robots. In this work we answer this question for all five pivoting models for hexagons and squares. Specifically, we answer it positively for the hexagonal monkey model by giving a universal reconfiguration algorithm in Section 2. For all other models we show that it is PSPACE-hard to determine whether we can reconfigure one configuration to another. In the process, we prove a stronger PSPACE-hardness result about a restricted form of motion planning with reversible, deterministic gadgets from [9] (our reduction highly limits the direction in which each edge can be traversed, effectively reducing the number of cases to consider). This framework has already proven useful in other swarm robot motion planning models [6, 8] and we believe the improvements here will aid in future PSPACE-completeness proofs. The framework is described in Section 3.2 and is used afterwards for hexagonal restricted robots in Section 3.3, and for all square models in Sections 3.4 and 3.5. Table 1 summarizes our results.

Table 1 Summary of results. The leapfrog move does not make sense for hexagonal modules.

Model	Restricted	Leapfrog	Monkey
Hexagons	PSPACE-hard (Thm. 14)	N/A	$O(n^3)$ universal (Thm. 13)
Squares	PSPACE-hard (Thm. 16)	PSPACE-hard (Thm. 17)	PSPACE-hard (Thm. 17) $O(n^2)$ if +5 modules [1]

2 Polynomial Algorithm for the Hexagonal Monkey Model

This section describes an algorithm that computes a sequence of $O(n^3)$ moves in the monkey model that transforms a given configuration with n modules into another. Our approach uses a **canonical configuration** defined as the configuration with n modules whose contact graph is a path and each module is only adjacent to modules above and/or below it. Since each move is reversible, an algorithm that takes a configuration and transforms it into the canonical configuration within $O(n^3)$ moves can be used to compute $O(n^3)$ moves between any pair of configurations. The main strategy is to increase the connectivity of the contact graph[1]. Note that if the contact graph is 2-connected, every convex corner of the configuration is movable, including the modules that are extremal in a grid direction. Then, there is a module that can move to become the new topmost module by attaching itself to a previous topmost module. We proceed in this manner inductively building the canonical configuration.

Definitions and Preliminaries. The contact graph G is the adjacency graph of the modules in a configuration. Since connectivity is important for the problem, we use the **block tree** $\mathcal{B}$ of the contact graph G. A graph is **2-connected** if it contains no **cut vertices**. A **block** (also 2-connected component) of G is a maximal subgraph of G that is 2-connected. We call a block containing a single edge a **trivial block**. We define $\mathcal{B}$ to be a bipartite tree whose nodes are the cut vertices of G in one partite set, and its blocks in the other partite set. There is an edge between two nodes if the corresponding cut vertex is contained in the corresponding block. The deletion of a cut vertex v of a connected graph G splits it into two or more components. A subgraph induced by such a component union with $\{v\}$ is called a **split component** of v. Similarly, a **2-cut** is a pair of vertices $\{v_1, v_2\}$ whose deletion increases the number of components of G. Its **2-split components** are the subgraphs induced by $\{v_1, v_2\}$ united with each of the components obtained by the deletion of $\{v_1, v_2\}$.

We now give some more specific definitions used in the algorithm. Note that a module corresponds to a vertex in G. We refer to them interchangeably. We label the topmost rightmost module of G the **root**. We root $\mathcal{B}$ at the node containing the root module. A cut vertex (2-cut) defines one **parent** (2-)split component, containing the root module, and one or more **child** (2-)split components. Such a cut vertex (2-cut) is called the **parent** of its children (2-)split components. A 2-split component ℓ is **trivial** if $|V(\ell)|$ is 3 or 4. The parent of such a component is also called trivial. Note that because G is a subset of the triangular grid, one of its faces is either the external face (whose edges form the **boundary**), a triangle, or encloses an empty position of the grid, which we call a **pocket**. In a 2-connected block ℓ, if v is a vertex in the boundary of ℓ and it is not incident to any pocket, deleting v can cause *only adjacent* vertices to become cut vertices. We call a 2-cut $\{v_1, v_2\}$ **adjacent** if v_1 and v_2 are adjacent. Note that when $\{v_1, v_2\}$ is an adjacent trivial 2-cut, the faces of the trivial 2-split component are triangles. Our algorithm uses the following fact about adjacent nontrivial 2-cuts.

▶ **Observation 1.** *If $\{v_1, v_2\}$ is an adjacent nontrivial 2-cut, then $\{v_1, v_2\}$ has only two 2-split components. Furthermore, if v_1 is movable, $\{v_1, v_2\}$ is the only 2-cut containing v_1.*

[1] Increasing connectivity of the contact graph of the configuration is a concept that has recently proven useful in the different setting of reconfiguring **sliding squares** [13].

The previous observation comes from the maximum degree of the triangular grid. For an adjacent 2-cut to have three 2-split components, two of them must be trivial. The fact that v_1 needs 3 adjacent empty positions around it to be movable implies that v_1 must be adjacent to 2 modules other than v_2. Any cycle through v_1 connecting the (2-)split components of $\{v_1, v_2\}$ must go through the two modules adjacent to v_1 that are not v_2.

The main technical part of our algorithm is a procedure called Merge that increases the 2-connectivity of G, i.e., decreases the number of nodes in $\mathcal{B}$. For that, we want to move modules in order to create new paths between blocks of G without destroying previously existing blocks. We define a **2-free** module to be a movable module whose movement preserves 2-connectivity in the block containing it (a module that is not in a 2-cut). A **crew** $c = (m_1, \ldots, m_k)$ is a sequence of modules that induce a connected component of G such that m_1 is 2-free, and m_i, $i \in \{2, \ldots, k\}$ is 2-free after the deletion of all m_j, $j \in \{1, \ldots, i-1\}$. For a given 2-connected subgraph ℓ of G, let $\overline{\ell}$ be the induced subgraph of G given by $V(G) \setminus V(\ell)$. A **bridge** from ℓ is a set of modules that were previously a crew that moved to create a path between ℓ and $\overline{\ell}$, thus potentially not being 2-free anymore. We say a set of modules **bridges from** ℓ if they move to create a new path between ℓ and $\overline{\ell}$. One of the goals of the algorithm is to get a crew of size three in a group of grid positions called a **flower** that is otherwise empty. That allows us to maneuver the modules in the crew to create a bridge while not creating new blocks. Let a **flower** be a set of grid positions defined by a **center** cell and the six adjacent positions. A flower is **adjacent** to a grid position if the flower does not contain it but contains a grid position that is adjacent to it. A flower is **valid** for a 2-connected configuration ℓ and a disjoint crew c if it contains *exactly* the modules in c (all modules in c and no other modules), and is adjacent to a module in ℓ.

The following are definitions that help us describe positions in the configuration. We might reflect and/or rotate the configuration in order to fit our description w.l.o.g., and the following definition always refer to the current frame of reference. A **row** containing a position p is the set of all positions $p + (-\frac{\sqrt{3}}{2}, \frac{1}{2})i$ for some integer i. An **ascending** (**descending**) path in a row ρ is a path $(m_1, \ldots, m_k)$ induced by modules in ρ such that m_{i+1} is the top-left (bottom-right) neighbor of m_i. An **extreme path** is a path induced by modules that are on the convex hull. Due to the geometry of the grid, extreme paths can only have six possible directions. A **SW extreme path** of a configuration ℓ is an ascending or descending path in the lower hull of ℓ. Given a position p in the grid, we use a sequence of arrow superscripts on p to describe positions nearby. For example, $p^{\uparrow\nearrow}$ refers to the position to the top-right of the position above p, i.e., $p + (\frac{\sqrt{3}}{2}, \frac{3}{2})$. We overload this notation to refer to the current positions of modules, replacing p by a module.

Main algorithm. We split the contact graph into two parts: the canonical path P which is a canonical configuration, and the remainder of the graph G. We initialize G to be the entire contact graph and P to be empty. Let $\mathcal{B}$ be the block tree of G rooted at the block containing the topmost rightmost module. We divide our algorithm into three phases. **Phase 1** is a prepossessing procedure that eliminates all trivial leaves of $\mathcal{B}$. Then, assume that every leaf of $\mathcal{B}$ contains at least three modules and no further procedure will change that. **Phase 2** transforms G into a 2-connected graph. While $\mathcal{B}$ is not a single node, let ℓ be a leaf of $\mathcal{B}$. We will apply Merge(ℓ), outlined in Algorithm 1, that will cause ℓ to merge with other nodes of $\mathcal{B}$ until G becomes 2-connected. **Phase 3** builds P. We decrease the size of G while adding modules to P by moving a crew on its boundary so that each of its members in turn move to become the new topmost module in the contact graph. We use a slightly modified version of Merge to produce such a crew without breaking the 2-connectivity of G.

2.1 Phase 1: Removing Trivial Leaves

Phase 1 reconfigures a connected configuration into one without vertices of degree 1 (which are in trivial leaves) in G. There are configurations in which it is not enough to just pivot the degree-1 modules, i.e., this task requires coordination with other modules. See Figure 4.

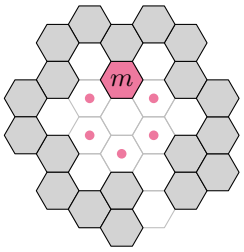


Figure 4 Configuration with one trivial leaf (m) that cannot be removed by pivoting it.

▶ **Lemma 2.** *A connected configuration of $n > 2$ hexagons can be transformed in $O(n^2)$ moves into a configuration without trivial leaves in the contact graph without breaking connectivity.*

Proof sketch. Let m be a degree-1 module. If it is possible to move m to a place where it is adjacent to more than one modules, then we do so. Else, we move m so that its shortest path to the root module is maximized. The full proof uses a detailed case analysis to show that, because of the specific position chosen for m, there is a nearby movable module with which m can coordinate to locally reduce the total number of new trivial leaves. ◀

2.2 Phase 2: Merging Leaves

The goal of Phase 2 is to take a connected configuration with no degree 1 vertices, and transform it into a 2-connected configuration in $O(n^3)$ moves. The main technical tool of this phase is the Merge procedure, outlined in Algorithm 1, which allows us to reduce the number of 2-connected components by merging them. Its input is a child (2-)split component of a cut vertex v (adjacent 2-cut $\{v_1, v_2\}$). We first apply the necessary rotations so that v ($\{v_1, v_2\}$) is farthest from the row ρ_0 containing the extreme SW path of ℓ. We then assume that ρ_0 does not include v ($\{v_1, v_2\}$) and neither does the row above it except for the base case when $|V(\ell)| = 3$ and ℓ is a split component, or when $|V(\ell)| = 5$ and ℓ is a 2-split component. The output of the algorithm is a set of modules that, after $O(|V(\ell)|^2)$ moves, bridges from ℓ.

Refer to Algorithm 1. Merge uses several other sub-procedures which we outline here. We call m the **ascending** module, which by its definition in line 2 is movable. It is either 2-free, in which case we will try to move it by cw pivots to its highest possible position in ρ_0 before it leaves ℓ; or it is part of a 2-cut, in which case we make it 2-free using sub-procedures. The end goal is to either bridge using m while it ascends in ρ_0 if it gets blocked by a vertex $m^* \notin \ell$, or accumulate 2-free modules at the top of the configuration where a valid flower will form. Then, the Bridge sub-procedure moves the valid flower around ℓ until it hits $\overline{\ell}$ where we create a bridge with the crew. There are three main Cases given by lines 4, 16 and 21.

Assume we are in Case 1. If m is in a trivial 2-cut, it will try to move up as explained before. Let m' be the module at $m^{\nearrow}$. By Observation 1, $\{m, m'\}$ is the only 2-cut containing m. If m succeeds in moving up, at least one unit, that leaves m' a cut vertex. Then, in line 7, we move the (up two) modules that are in the 2-split component of $\{m, m'\}$, restoring 2-connectivity. During m's ascension in ρ_0, we identify whether a valid flower gets formed. In the positive case, Bridge will accomplish our goal. During its ascension, m might be blocked

by a module $m^* \in \overline{\ell}$. If certain conditions are satisfied, the Local-Bridge sub-procedure uses m to create a bridge to m^*. Else, the Incorporate sub-procedure moves m to the row ρ_1 above it, or out of ℓ, and we can find a new ascending module.

Now assume that m is part of a nontrivial 2-cut (Case 2). Then, either m is part of an adjacent 2-cut or it is incident to a pocket. In the case m or an adjacent module is part of an adjacent 2-cut we recurse in the child 2-split component, which makes m 2-free. Else (Case 3), we either use Deflate, which decreases the number of empty positions enclosed by the pocket, or Bubble-Up, which moves one of such empty positions up. In some situations, Deflate produces a 2-free module in ρ_0 that will be the next ascending module.

Algorithm 1 Merge(ℓ).

```
1  while True do
2      Let m be the topmost module in a SW extreme path of ℓ;
3      Let ρ_{-1}, ρ_0 and ρ_1 be the rows below, of, and above m respectively;
4      if m is 2-free or part of an adjacent trivial 2-cut then
5          Pivot m cw to the highest position in ρ_0 before it leaves ℓ;
6          if m was part of a trivial 2-cut then
7              Pivot cw once the other modules in the trivial child;
8          if m is ever in a crew c of size 3 in a valid flower F during its ascension then
9              Return Bridge(F, ℓ − c);
10         else if m bridges from ℓ then
11             Return m;
12         else if the requirements of Local-Bridge(m) are met then
13             Return Local-Bridge(m);
14         else
15             Incorporate(m);
16     else if {m, m^↗} or {m^↘, m^↘↗} is a nontrivial 2-cut then
17         Let ℓ' be the child 2-split component of the highest such 2-cut;
18         c' := Merge(ℓ');
19         if c' bridges between ℓ and ℓ̄ then
20             Return c';                          ▷ c' already merges ℓ into another block.
21     else
22         Deflate(m^↗) or Bubble-Up(m^↗);                              ▷ m^↗ is empty
23     end
24 end
```

Bridge(ℓ, F). The operation takes a 2-connected ℓ and a valid flower F containing a crew $c = (m_1, m_2, m_3)$ where m_1 was an ascending module. It returns c after a sequence of moves that transforms c into a bridge from ℓ. Compute a maximal sequence of flowers $(F_1 = F, \ldots, F_k)$, where each subsequent flower is adjacent to ℓ, containing no modules except for c, and obtained by moving the previous flower by one grid unit around the boundary of ℓ. We choose to move cw or ccw around ℓ based on the following condition. If ℓ has a parent cut vertex, then choose arbitrarily. Else, if ℓ has a parent adjacent 2-cut $\{v_1, v_2\}$ where v_1 is movable, we chose the direction towards v_2 so that F_k is not adjacent to v_1. Since G is connected and planar, and there are vertices in $\overline{\ell}$, F_k is adjacent to a module m^* in $\overline{\ell}$. We show how to compute the sequence of moves to bring the the crew with the sequence of flowers $(F_1, \ldots, F_k)$ and finally bridge between ℓ and m^* in F_k.

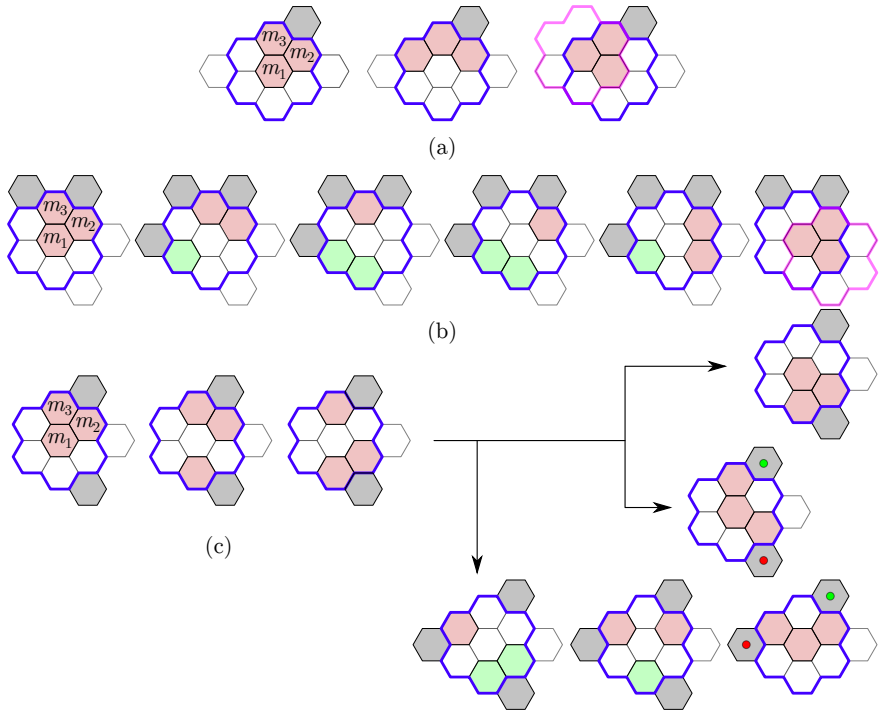


Figure 5 Maneuvers used to rotate around m_1 a crew that induces a cycle. A possible next flower is shown in pink.

If the modules in c induce a connected graph, this graph is either a triangle, a straight path or a "bent" path. A configuration of c in a valid flower is **useful** if a module of c is adjacent to ℓ and c induces a triangle or a "bent" path with m_1 in the center of the flower (i.e., both endpoints are adjacent to modules outside the flower). We show how to reach every useful configuration of c in a valid flower F_i. This is enough to accomplish our objective since: (i) by definition, $F_i \cap F_{i+1}$ is adjacent to ℓ and there is a useful configuration contained in the intersection of F_i and F_{i+1} (Figure 5 (a)–(b)); and (ii) if $m^- \in \ell$ is the only module adjacent to F_k and m^* is the only module of $\overline{\ell}$ adjacent to F_k and they are across from the center of F_k, e.g. m^- (m^*) is at the topmost (bottommost) position adjacent to F_k, then we can move F_k one more unit along the boundary of ℓ, contradicting the maximality of the sequence. By (i) we can transition between flowers F_i and F_{i+1} through a useful configuration, making both valid. By (ii) there is a useful configuration at F_k that bridges between m^- and m^*.

In the full version [2], we present four **maneuvers** shown in Figure 5 along with omitted proofs. Note that, by the fact that c is a crew, we have a guarantee that some positions adjacent to the flower are empty. We then use them to show the following lemma.

▶ **Lemma 3.** *Every useful configuration of a crew c in a valid flower can be reached from any useful configuration.*

▶ **Lemma 4.** *Bridge(ℓ, F) performs $O(|V(\ell)|)$ moves and bridges from ℓ while not breaking connectivity. After its execution, ℓ is still 2-connected. If $\{v_1, v_2\}$ is the parent 2-cut of ℓ and v_1 is movable, then v_1 remains movable.*

Deflate(p) and Bubble-Up(p). These operations take an empty position p to the top-right of a module m which is a corner of a 2-connected subgraph ℓ of the contact graph. We assume that m is a corner of ℓ in its SW extreme path, i.e., $m^{\nwarrow}$, $m^{\swarrow}$, and $m^{\downarrow}$ are empty.

Then, p is enclosed by ℓ by 2-connectivity. Refer to Figure 6. Deflate requires that positions surrounded by a red line in Figure 6 (a) or (c) are as shown. In particular, if $m^{\searrow\nearrow}$ is full, then $m^{\uparrow\nwarrow}$ is empty. Then, the operation fills p with a module adjacent to m and preserves 2-connectivity of ℓ, effectively reducing its area. Bubble-Up requires that positions surrounded by a red line in Figure 6 (b) are as shown. In particular, if $m^{\searrow\nearrow}$ is full, then $m^{\uparrow\nwarrow}$ is full. We additionally require that $\{m^{\searrow}, m^{\searrow\nearrow}\}$ is not a nontrivial 2-cut. Then, the operation moves the empty position and m to their top-left position while preserving 2-connectivity.

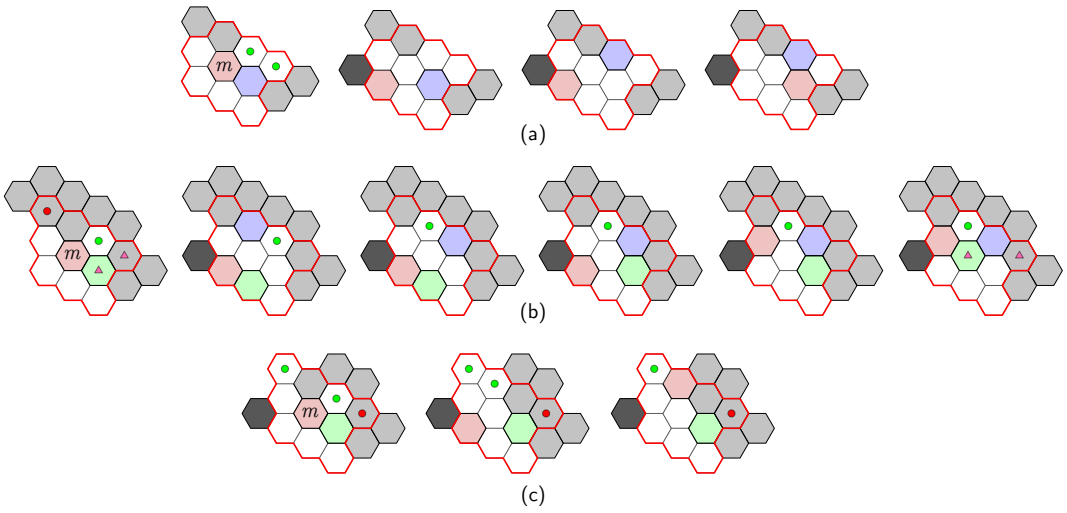


Figure 6 Operations used in Deflate(m_1).

Figure 6 shows the operations assuming that m performs a monkey move by pivoting cw.

▶ **Lemma 5.** *Deflate(p) and Bubble-Up(p) perform $O(1)$ moves, do not break connectivity, and the resulting ℓ is 2-connected.*

Shift(M). Although not used directly in Merge, this operation is used in following subprocedures. The input is an ascending or descending path of modules $M = (m_1, \ldots, m_t)$ in the same row ρ and in the boundary of G. We require that none of the modules are cut vertices, m_1 is movable, and $m_i^{\downarrow}$ is empty $\forall i \in \{1, \ldots, t-1\}$. We describe the operation for descending M (Figure 7). There are two cases. In the first case, no module $m_i \in M$ is such that $m_i^{\nearrow}$ is empty and $m_i^{\nearrow\searrow}$ is either empty or contains a module only adjacent to M. Then, move each m_i cw from $i = 1$ to t (Figure 7 (d) to (f)). In the second case, let $m_i \in M$ be the first module such that $m_i^{\nearrow}$ is empty and $m_i^{\nearrow\searrow}$ is either empty or contains a module m' only adjacent to M. Move all m_j from $j = 1$ to $i - 1$ and m' (if it exists) by pivoting cw. Then, apply Deflate(m_i) and move back all the m_j and m' to their original positions by a ccw pivot. This vacates m_i's original position (Figure 7 (a) to (d)). If $i \neq t$, apply Shift recursively on $(m_{i+1}, \ldots, m_t)$.

In the full version [2] we prove the following statement.

▶ **Lemma 6.** *Shift(M) performs $O(|M|)$ moves, does not break the connectivity and, after it terminates, all pockets of ℓ remain intact except for possibly one that has m_t in its boundary.*

Inflate(m). This operation uses Shift to make a concave corner convex, possibly creating a new empty space enclosed by ℓ. This will be used in Local-Bridge. The input m is an ascending module in a 2-connected ℓ. We require that $m^{\uparrow}$ and $m^{\nearrow}$ are full, neither

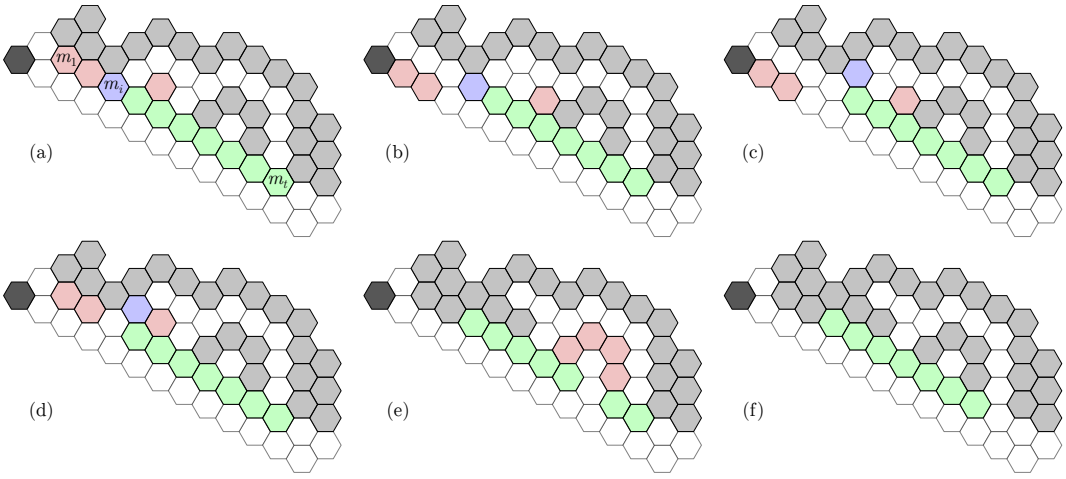


Figure 7 Illustration of Shift(M) where M is descending.

$\{m^{\uparrow}, m^{\uparrow\nearrow}\}$ nor $\{m^{\uparrow\uparrow}, m^{\uparrow\uparrow\nwarrow}\}$ are in adjacent nontrivial 2-cuts, and that at least one position in $\{m^{\uparrow\nwarrow}, m^{\uparrow\nearrow}, m^{\uparrow\uparrow\nwarrow}\}$ is full. Inflate moves the module at $m^{\uparrow}$ to $m^{\nwarrow}$ via a series of operations, returning such module.

Refer to Figure 8 (a)–(c) for examples (or [2] for a full proof). In short, we use Shift to move away modules adjacent to the blue module, so that we can move it out. Then, we reverse the Shift operation to put the moved models, except for the blue one, into their original place.

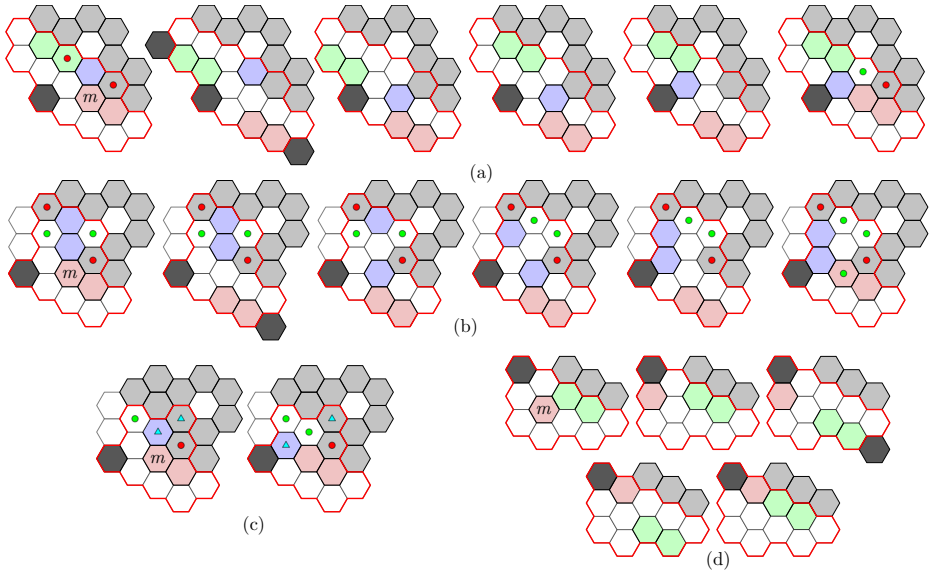


Figure 8 Operations used in Local-Bridge.

▶ **Lemma 7.** *Inflate*(m) *performs* $O(|V(\ell|)$ *moves, does not break connectivity and preserves 2-connectivity of* ℓ.

Local-Bridge(m). This operation is used when there is an opportunity to create a bridge when m either gets blocked by m^* on its way to the top of ρ_0 or it reaches the top and it would jump to $\overline{\ell}$. We require that, if a cw pivot brings m to ρ_{-1}, then $m^{\nearrow}$ is full and at

least one position in $\{m^{\uparrow\nwarrow}, m^{\uparrow\nearrow}, m^{\uparrow\uparrow\nwarrow}\}$ is full. We recurse on a child component, calling Merge if there is an adjacent nontrivial 2-cut forbidden by Inflate. That guarantees that we can apply Inflate which would create a bridge from ℓ. If a cw pivot maintains m in ρ_0, then it would land on a module $m^* \in \overline{\ell}$. We require that the maximal ascending path M ending in $m^{\uparrow}$ can be shifted down by Shift(M), i.e., ρ_0 must contain only m below M; see Figure 8 (d). Then, we "squeeze" m in the space between m^* and M creating a bridge. We do that by moving m out of ℓ, Shift M down, moving m back and Shift M back.

▶ **Lemma 8.** *Local-Bridge(m) bridges from ℓ, does not break connectivity, and preserves 2-connectivity of ℓ. It uses $O(|V(\ell)|) + T_m(|V(\ell')|)$ moves where $T_m(|V(\ell')|)$ is the number of operations performed by Merge in ℓ'.*

Incorporate(m). Whenever a local bridge was not possible, this operation either incorporates m into ρ_1 or leaves m attached to a module in $\overline{\ell}$ with the promise that some module will ascend in ρ_0 and bridge (Figure 9 (d)). There are four cases. In case 1, we check if we can call Deflate at position $m^{\uparrow\nearrow}$. In the positive case, we move m and a possible neighbor m' in ρ_0 out of the way, call Deflate, and move m and possibly m' back (Figure 9 (a)–(b)). In case 2, $m^{\nearrow}$ is empty and $m^{\uparrow\nearrow}$ is full. Then, we "squeeze" m into $m^{\nearrow}$ by using Shift operations, similar to Bridge (Figure 9 (c)). In case 3, if we pivot m cw, that brings m to ρ_1 and makes its degree 1. Then, we apply some local movements in order to incorporate m into ρ_1 while maintaining 2-connectivity (Figures 9 (e) and (g)). In case 4, we are not in the previous cases and we simply pivot m cw. Note that m might leave ℓ (Figure 9 (d)). We explore this case from now. As shown in the proof of Lemma 9, there is a guarantee that a subsequent module s in ρ_0 will ascend. There are three possible cases, either (i) s creates a bridge using m (as in Figure 9 (d)), in which case nothing needs to be done; (ii) s calls Local-Bridge or Bridge. Then, pivot m twice counterclockwise before Bridge or after Local-Bridge; or (iii) s calls Incorporate. Then, there is either another module in ρ_0 or we can move s back to ℓ and apply Local-Bridge.

▶ **Lemma 9.** *Incorporate(m) uses $O(|V(\ell)|)$ moves and brings m to ρ_1 in every situation that m would go to ρ_{-1} by pivoting cw to which Local-Bridge does not apply. It does not break connectivity and maintains 2-connectivity of ℓ. Any created degree-1 module outside ℓ can be reincorporated in ℓ, thus, no new block is created.*

Analysis.

▶ **Lemma 10.** *If $\ell \neq G$ is a leaf block of $\mathcal{B}$, Merge(ℓ) performs $O(|V(\ell)|^2)$ moves merging ℓ and a subset of nodes of $\mathcal{B}$ into a single block while not creating any other new blocks.*

Proof sketch. A key observation is that each section of the perimeter can only be traversed by at most three ascending modules until either a local bridge or a valid flower is formed. Every time we use Incorporate to hide a module in ρ_1 we have the guarantee that, if either the next or the next two ascending modules reaches m, then Local-Bridge or Bridge will be called and the method terminates. We can then charge the moves of a module to the perimeter. Hence, each level of recursion of Merge makes a linear number of moves. Another key observation is that there are only a constant number of recursive calls. Since we always recurse on a smaller problem, the upper bound on the number of moves is $O(|V(\ell)|^2)$. ◀

▶ **Corollary 11.** *G can be made 2-connected in $O(n^3)$ moves.*

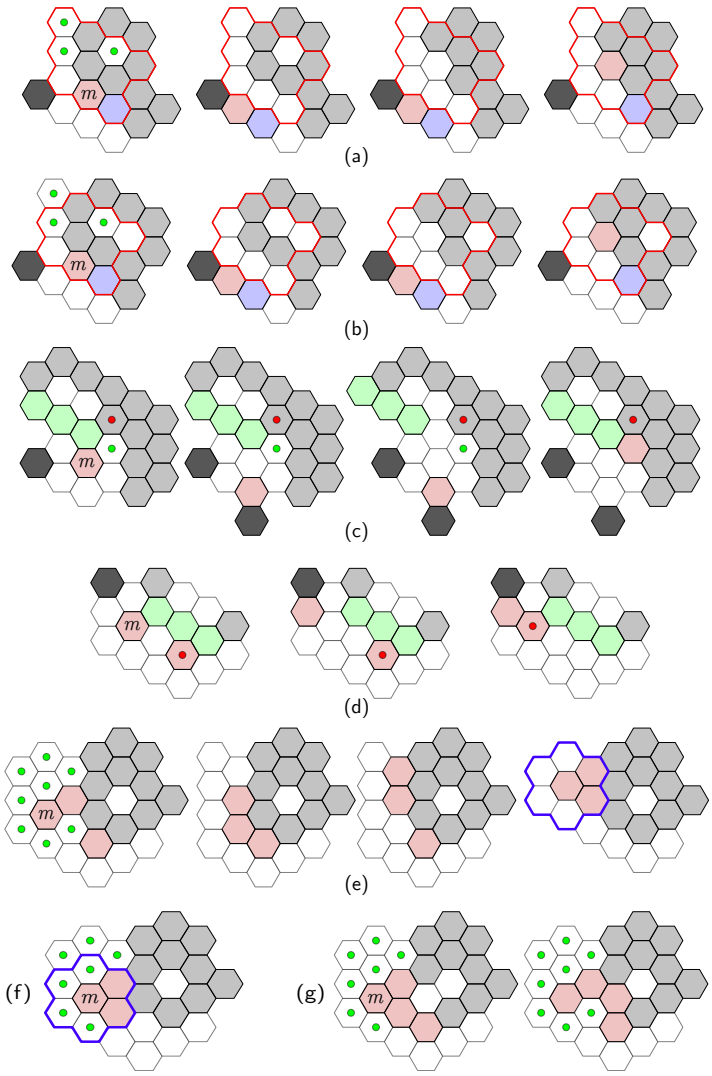


Figure 9 Operations used in Incorporate.

2.3 Phase 3: Building the Canonical Path

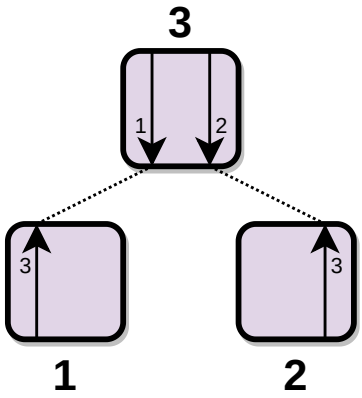
In the final phase, we will show that once the configuration is 2-connected, we can start moving modules onto the end of our path P at a cost of $O(n^2)$ moves per module.

▶ **Lemma 12.** *If G is 2-connected, in $O(n^2)$ moves we can produce a 2-free module on an extreme path of G while maintaining the 2-connectivity of G.*

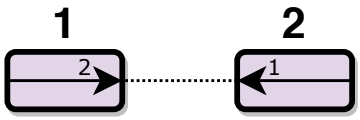
Proof. We apply a subset of operation Merge to G. Then, this proof becomes a special case of Lemma 10, where $\ell = G$. In Merge, our goal is to bridge between ℓ and $\overline{\ell}$ maintaining ℓ 2-connected. Here, $\overline{\ell}$ is empty and Local-Bridge will never be called since there are no obstacles for ascending modules. Then, we are always able to produce a crew in an extremal position. Moving the crew to P does not affect 2-connectivity of G by definition. ◀

Our main theorem is a direct consequence of Lemma 2, Corollary 11 and Lemma 12.

▶ **Theorem 13.** *Any connected configuration of n hexagonal modular robots can be reconfigured to any other with $O(n^3)$ pivoting moves in the Monkey model, while maintaining connectivity.*



(a) The locking 2-toggle gadget (L2T).



(b) The 1-toggle gadget.

Figure 10 Examples of reversible, deterministic gadgets. Purple boxes are states of the gadget, labeled with a number outside the box. Arrows represent transitions from one location to another. The small number close to an arrow indicates the state obtained by the transition. Dotted lines help visualize which states are connected by transitions in the gadget.

3 PSPACE-hardness Reductions

In this section we show PSPACE-hardness for all other models. Our reduction follows the framework introduced in [9]. We reduce from a reachability problem: given an agent that moves along a graphlike structure whose traversability changes in response to the agent's actions, is there a series of moves which takes the agent from a start to a target location.

▶ **Theorem 14.** *Given two configurations of n hexagonal modules, it is PSPACE-hard to determine if we can reconfigure from one to the other using only restricted moves.*

In Section 3.1 we describe the reachability problem introduced in [9] and the pieces we need to simulate to create the reduction. We introduce a few modifications to this problem and show it remains PSPACE-hard in Section 3.2. In Section 3.3 we discuss how to simulate each of the gadgets with hexagonal modules. Reductions for other models are in Section 3.4.

3.1 Preliminaries

We reduce from a variation of 1-player motion planning with the locking 2-toggle (L2T) [9]. This restricted variant is called *1-toggle-protected motion planning with the locking 2-toggle* and described in Section 3.2. In the **1-player motion planning** problem we want to decide whether an agent has a series of moves that will take it to a target location. The constructs we use in this problem are **gadgets** which have **locations** (entrances and exits), **states**, and **transitions**. The agent is always at some unique location. Transitions are an ordered pair of state and location pairs. If an agent is at some specific gadget location and the gadget is in a state matching the first pair, then the agent can move to the location in the second pair which changes the state of the gadget to the state in the second pair (see Figure 10). A **system of gadgets** is a set of gadgets and **connections** between locations in those gadgets. The agent can freely move between locations that have connections. Some gadget transitions form a matching – we call these matched pairs **tunnels**.

In order for us to reduce from this problem, we need to use modules to represent the agent, the gadgets (specifically a locking 2-toggle and a branching hallway gadget), connections between locations, and a goal location. In order to reduce from 1-toggle-protected motion planning with the locking 2-toggle we need to create the following constructions:

- **Wires** which allow the modules to travel between parts of the configuration. These simulate the connection graph edges that allow the agent to travel between locations.
- **Branching hallways** which connect three wires together and allow the modules to travel down any of them.
- **Locking 2-toggle** which is a 3 state, 4 location gadget shown in Figure 10a. The gadget has two tunnels which are both traversable in state 3. After taking either transition, the only option is returning back and restoring the gadget to its prior state.
- **Win gadget** which can only be reconfigured if two additional modules reach it, simulating the goal location in the motion planning problem.

3.2 1-toggle-protected Motion Planning

In this section we strengthen the result from [9] to show that motion planning with reversible, deterministic gadgets with interacting tunnels is PSPACE-complete even when connections can only be traversed as though they are 1-toggles. We will consider only branchless systems of gadgets, but we will allow the branching hallway gadget. In a **branchless** system of gadgets, the connections between locations form a matching [4]. The **branching hallway gadget** is a 1-state, 3-location gadget with traversals among all three pairs of locations.

An instance of **1-toggle-protected motion planning** with a set of gadgets G is an instance of branchless 1-player motion planning with G as well as the branching hallway gadget and the 1-toggle, where one end of every connection is a location on a 1-toggle. Intuitively, this requires that every edge in the connection graph acts as a 1-toggle.

▶ **Theorem 15.** *1-toggle-protected planar 1-player motion planning problem with a reversible, deterministic, on-tunnels gadget with interacting tunnels is PSPACE-complete.*

3.3 Reduction for Hexagonal Modules

We now focus on describing how to simulate each of the pieces with hexagonal modules. The agent is represented by two modules and while these could go different ways, our instance contains several obstacles that can only be crossed by two modules working together. For simplicity we refer to the two modules that form the agent as *the agent modules* (shown in orange in the figures).

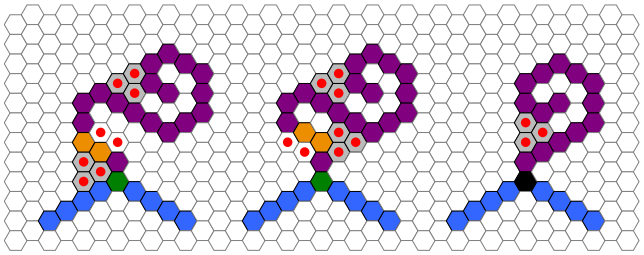


Figure 11 Each corner contains a spiral that prevents it from moving. In the protected case, the agent can go to a specific location allowing two other modules on the other side of the spiral to move (left and center), while in the blocked case, the spiral cannot be crossed at all (right).

We simulate wires with sequences of modules in line segments. We also need to be able to turn without letting corner modules move. We simulate these turns using two types of corners: **protected** and **blocked** (Figure 11).

The branching hallway, shown in Figure 12, allows an agent that arrives on any of the three wires to leave on any of the other two. This construction acts as a branching hallway with 1-toggles on two of its wires. We can implement the toggle on the third wire by adding two protected corners.

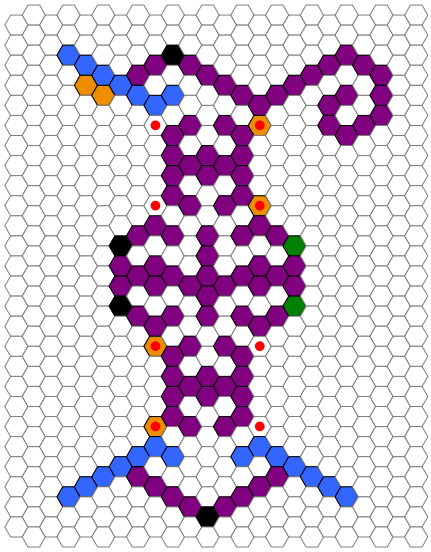


Figure 12 The branching hallway gadget simulated with hexagonal modules. Wires are shown in blue, and the two modules simulating the agent are shown in orange. Protected corners are drawn as green modules, while blocked corners are drawn as black. Both types of corners contain additional robots (omitted for clarity).

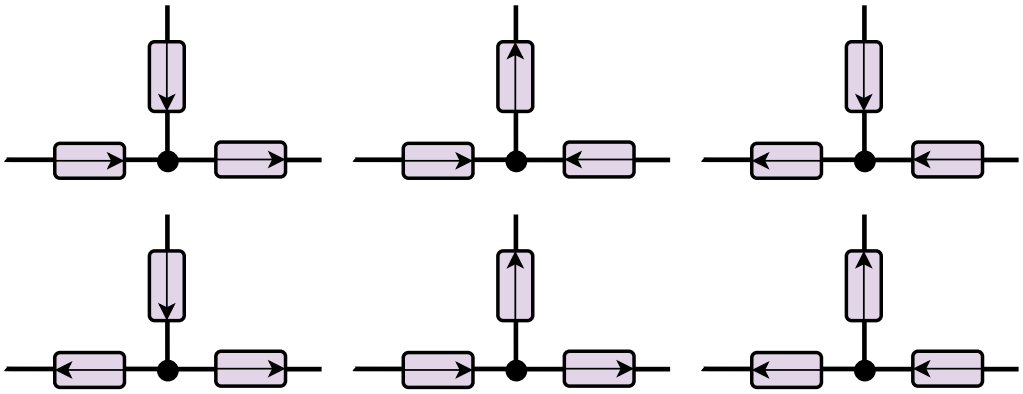


Figure 13 The six configurations of a branching hallway with endpoints connected to 1-toggles.

The other main gadget needed for the reduction is the locking 2-toggle shown in Figure 14.

3.3.1 Finishing steps

Proof. (of Theorem 14) Our reduction follows the framework in [9]. Given a problem instance for 1-toggle-protected motion planning with the locking 2-toggle, we embed in a way that all edges are drawn with polylines that are multiples of 60°, replacing gadgets with the

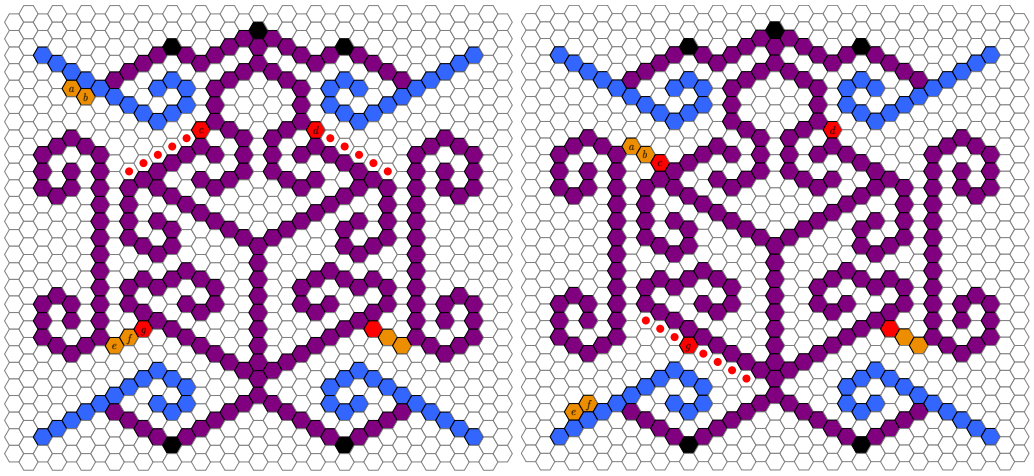


Figure 14 (left) The locking 2-toggle simulated with hexagonal modules (state 3 in Figure 10a). Other than the agent (a and b), the modules that can move are c and d (but not at the same time). (right) Once a, b and c form a bridge, they create a cycle allowing e, f and g to move. Two of the three modules can exit the gadget along the bottom wire. This changes the state of the 2-toggle (state 1 in Figure 10a).

corresponding module configurations (adding side switch and wire cut gadgets as needed, as well as 1-gaps to all wire segments). Finally, we place two additional modules at the initial position to define the agent. Since each gadget takes constant space, the problem instance will have polynomial size. Our goal configuration is the same configuration with only one change (the state of the win gadget).

If the problem instance is solvable, there is a way for the agent to reach the win gadget, change its state, and then return back to the initial position in the exact reverse path. By doing so we reset every gadget except the win gadget back to its original state. If the problem instance is not solvable, the agent cannot reach the win gadget and thus the reconfiguration problem will also be infeasible. ◀

3.4 Square Modules with the Restricted Move Model

▶ **Theorem 16.** *Given two configurations of n square modules, it is PSPACE-hard to determine if we can reconfigure from one to the other using only restricted moves.*

Our reduction is analogous to the hexagonal reduction. We quickly list the pieces and a small description for each, but for brevity the proof of correctness of each single gadget is removed. The arguments are analogous to the hexagonal counterpart and we present a full list of our gadgets in the full version [2].

The branching hallway gadget is show in Figure 15 and works like the hexagonal version.

The L2T gadget in the open state can be seen in Figure 16. Again, this gadget has exactly the same functionality as its hexagonal counterpart. The reduction works similarly and the proof for Theorem 16 follows a similar format as Theorem 14.

3.5 Hardness for the Square Model for Monkey and Leapfrog Models

Our final reduction applies to both remaining models for square modules.

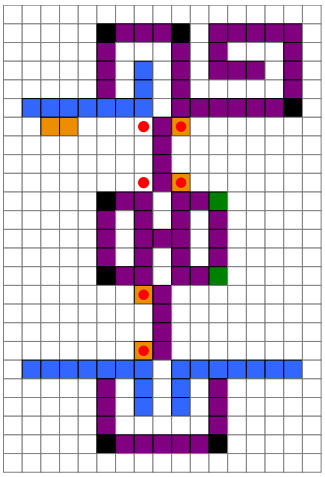


Figure 15 The branching hallway gadget for squares under the restricted model.

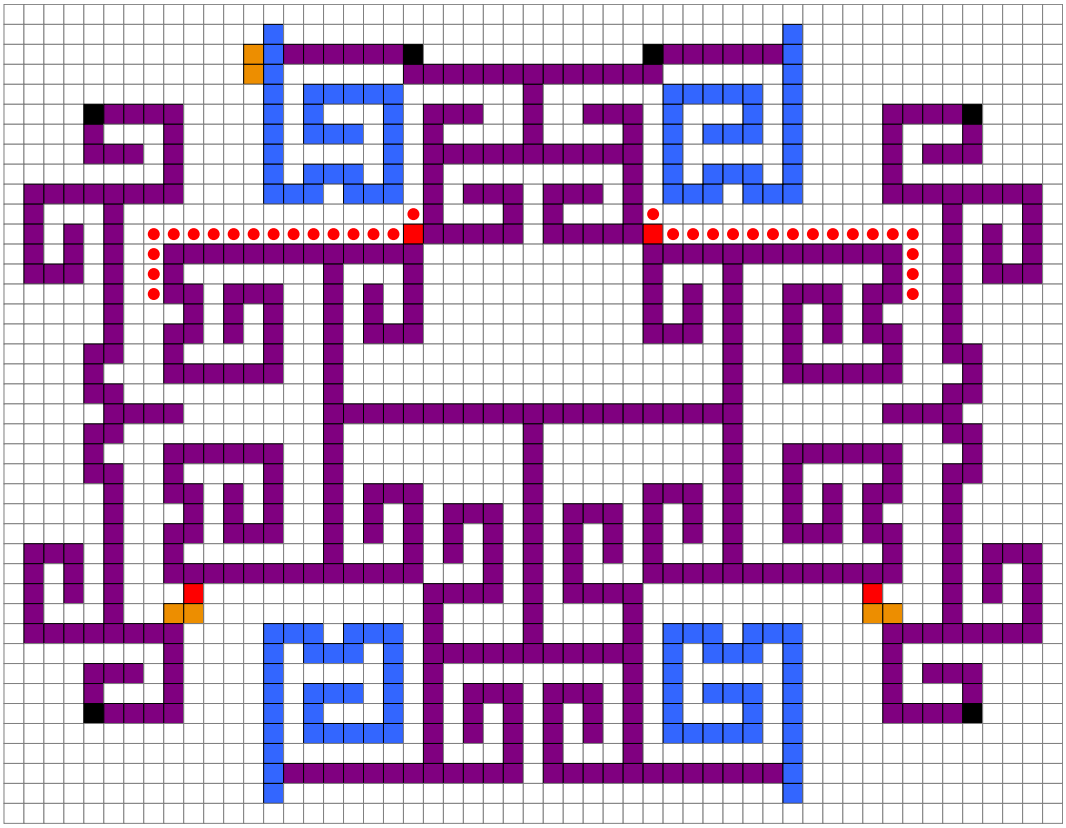


Figure 16 L2T in the open state. At this point the gadget can come from either of the top wires.

▶ **Theorem 17.** *Given two configurations of n square modules, it is PSPACE-hard to determine whether one can be reconfigured into the other in both the monkey and the leapfrog models.*

As before, the reduction is from 1-toggle-protected motion planning with the locking 2-toggle, but simpler. The main differences are as follows:

- A leapfrog move can pass through obstacles or bends without creating global cycles. All the cycles created by the agent module are local, with size at most 8, which allows us to have purely local arguments.
- Because of this change, we can now represent the agent with a single module. This eliminates the need to prove that multiple modules have to work together (and all other intricacies related to the case of a 2-module agent).
- Another interesting advantage is that we can represent a wire with two parallel sequences of modules (5 units apart). The agent will move between the two lines, which reduces the need of worrying about which side the agent is on.
- Finally, the reduction works for the leapfrog model, but even if we allow monkey moves the result holds. Thus, a single reduction will work for both models.

The gadgets we use are shown in Figure 17 and Figure 18. Due to space constraints, we defer the description and proof of correctness to [2].

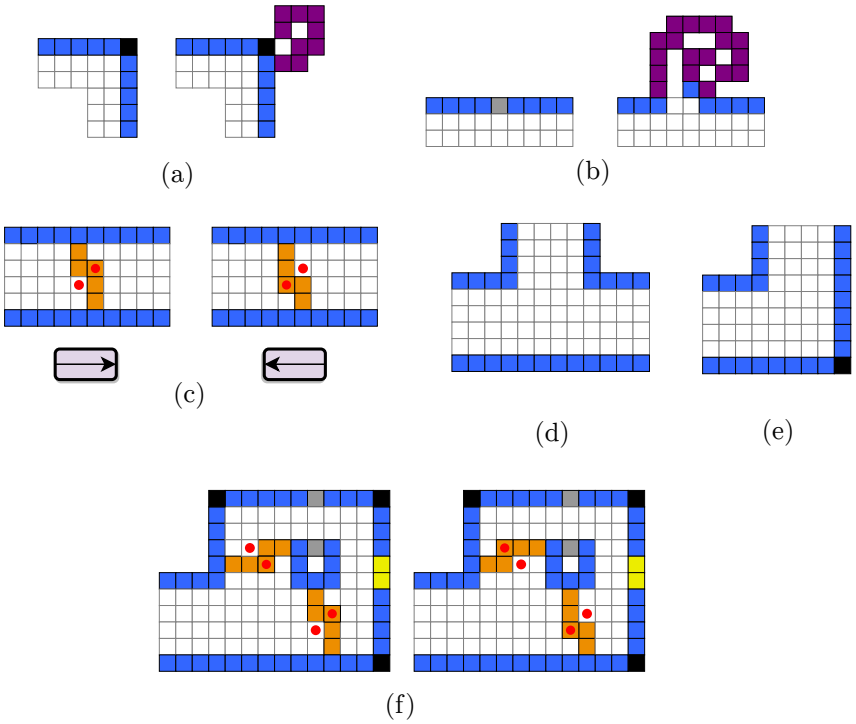


Figure 17 Gadgets used in PSPACE reduction (for leapfrog and monkey models).

4 Conclusions

This paper answers fundamental question, but also opens up further line of research. First, for hexagonal modules under the monkey model (where universal reconfiguration is possible), there is a gap between the upper bound of our algorithm (Theorem 13) and the naive $\Omega(n^2)$ lower bound (number of moves needed to transform a horizontal strip into a compact hexagon). Even if the gap is closed, then the interest would be to design a distributed algorithm and/or to consider a strategy that does many moves in parallel.

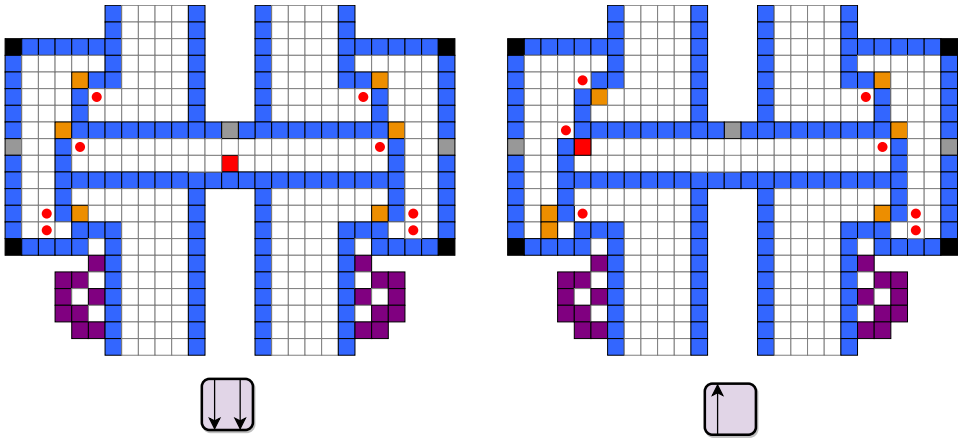


Figure 18 L2T gadget with square modules for the monkey model. In the figure two of the three possible states are shown (third one is symmetric).

For models in which universal reconfiguration is not possible it would be nice to find a local property that would allow reconfiguration between many configurations. For example, with square modules and the monkey operation, reconfiguration is possible as long as both configurations have five modules on the outer shell that can move (these modules are called *musketeers* [1]).

References

1 Hugo A. Akitaya, Esther M. Arkin, Mirela Damian, Erik D. Demaine, Vida Dujmovic, Robin Y. Flatland, Matias Korman, Belén Palop, Irene Parada, André van Renssen, and Vera Sacristán. Universal reconfiguration of facet-connected modular robots by pivots: The O(1) musketeers. *Algorithmica*, 83(5):1316–1351, 2021. doi:10.1007/s00453-020-00784-6.

2 Hugo A. Akitaya, Erik D. Demaine, Andrei Gonczi, Dylan H. Hendrickson, Adam Hesterberg, Matias Korman, Oliver Korten, Jayson Lynch, Irene Parada, and Vera Sacristán. Characterizing universal reconfigurability of modular pivoting robots. *CoRR*, abs/2012.07556, 2020. arXiv:2012.07556.

3 Greg Aloupis, Sébastien Collette, Mirela Damian, Erik D. Demaine, Robin Flatland, Stefan Langerman, Joseph O'Rourke, Val Pinciu, Suneeta Ramaswami, Vera Sacristán, and Stefanie Wuhrer. Efficient constant-velocity reconfiguration of crystalline robots. *Robotica*, 29(1):59–71, 2011. doi:10.1017/S026357471000072X.

4 Joshua Ani, Erik D. Demaine, Dylan H. Hendrickson, and Jayson Lynch. Trains, games, and complexity: 0/1/2-player motion planning through input/output gadgets. *CoRR*, abs/2005.03192, 2020. arXiv:2005.03192.

5 Nora Ayanian, Paul J. White, Ádám Hálász, Mark Yim, and Vijay Kumar. Stochastic control for self-assembly of XBots. In *Proc. ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (IDETC-CIE)*, pages 1169–1176, 2008. doi:10.1115/DETC2008-49535.

6 Jose Balanza-Martinez, Austin Luchsinger, David Caballero, Rene Reyes, Angel A Cantu, Robert Schweller, Luis Angel Garcia, and Tim Wylie. Full tilt: Universal constructors for general shapes with uniform external forces. In *Proc. 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2689–2708, 2019.

7 Nadia M. Benbernou. *Geometric Algorithms for Reconfigurable Structures*. PhD thesis, Massachusetts Institute of Technology, 2011.

8 David Caballero, Angel A. Cantu, Timothy Gomez, Austin Luchsinger, Robert Schweller, and Tim Wylie. Relocating units in robot swarms with uniform control signals is PSPACE-complete. In *Proc. 32th Canadian Conference on Computational Geometry*, 2020.

9 Erik D. Demaine, Dylan H. Hendrickson, and Jayson Lynch. Toward a general complexity theory of motion planning: Characterizing which gadgets make games hard. In *Proc. 11th Innovations in Theoretical Computer Science Conference (ITCS)*, volume 151, pages 62:1–62:42, 2020. doi:10.4230/LIPIcs.ITCS.2020.62.

10 Adrian Dumitrescu and János Pach. Pushing squares around. *Graphs and Combinatorics*, 22(1):37–50, 2006. doi:10.1007/s00373-005-0640-1.

11 Adrian Dumitrescu, Ichiro Suzuki, and Masafumi Yamashita. Motion planning for metamorphic systems: feasibility, decidability, and distributed reconfiguration. *IEEE Transactions on Robotics*, 20(3):409–418, 2004. doi:10.1109/TRA.2004.824936.

12 Robert Fitch, Zack Butler, and Daniela Rus. Reconfiguration planning for heterogeneous self-reconfiguring robots. In *Proc. 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 2460–2467, 2003. doi:10.1109/IROS.2003.1249239.

13 Irina Kostitsyna, Irene Parada, Willem Sonke, Bettina Speckmann, and Jules Wulms. Compacting squares. Manuscript, 2020.

14 Tom Larkworthy and Subramanian Ramamoorthy. A characterization of the reconfiguration space of self-reconfiguring robotic systems. *Robotica*, 29(1):73–85, 2011. doi:10.1017/S0263574710000718.

15 An Nguyen, Leonidas J. Guibas, and Mark Yim. Controlled module density helps reconfiguration planning. In *Algorithmic and Computational Robotics: New Dimensions (2000 WAFR)*, pages 23–36. A. K. Peters, 2001.

16 Cynthia Sung, James Bern, John Romanishin, and Daniela Rus. Reconfiguration planning for pivoting cube modular robots. In *Proc. 2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1933–1940, 2015. doi:10.1109/ICRA.2015.7139451.

Adjacency Graphs of Polyhedral Surfaces

Elena Arseneva
Saint Petersburg State University, Russia

Linda Kleist
Technische Universität Braunschweig, Germany

Boris Klemz
Universität Würzburg, Germany

Maarten Löffler
Utrecht University, The Netherlands

André Schulz
FernUniversität in Hagen, Germany

Birgit Vogtenhuber
Technische Universität Graz, Austria

Alexander Wolff
Universität Würzburg, Germany

Abstract

We study whether a given graph can be realized as an adjacency graph of the polygonal cells of a polyhedral surface in $\mathbb{R}^3$. We show that every graph is realizable as a polyhedral surface with arbitrary polygonal cells, and that this is not true if we require the cells to be convex. In particular, if the given graph contains K_5, $K_{5,81}$, or any nonplanar 3-tree as a subgraph, no such realization exists. On the other hand, all planar graphs, $K_{4,4}$, and $K_{3,5}$ can be realized with convex cells. The same holds for any subdivision of any graph where each edge is subdivided at least once, and, by a result from McMullen et al. (1983), for any hypercube.

Our results have implications on the maximum density of graphs describing polyhedral surfaces with convex cells: The realizability of hypercubes shows that the maximum number of edges over all realizable n-vertex graphs is in $\Omega(n \log n)$. From the non-realizability of $K_{5,81}$, we obtain that any realizable n-vertex graph has $\mathcal{O}(n^{9/5})$ edges. As such, these graphs can be considerably denser than planar graphs, but not arbitrarily dense.

2012 ACM Subject Classification Mathematics of computing → Graphs and surfaces; Mathematics of computing → Combinatoric problems

Keywords and phrases polyhedral complexes, realizability, contact representation

Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.11

Related Version *Full Version*: https://arxiv.org/abs/2103.09803

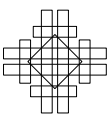
Funding *Elena Arseneva*: partially supported by RFBR, project 20-01-00488.
Boris Klemz: supported by DFG project WO 758/11-1.
Birgit Vogtenhuber: partially supported by the Austrian Science Fund within the collaborative DACH project *Arrangements and Drawings* as FWF project I 3340-N35.

Acknowledgements We thank the organizers of Dagstuhl Seminar 19352 "Computation in Low-Dimensional Geometry and Topology" for bringing us together. We are particularly indebted to seminar participant Arnaud de Mesmay for asking a question that initiated our research. We also thank the anonymous referees of our EuroCG 2020 submission for their helpful comments.

1 Introduction

A *polyhedral surface* consists of a set of interior-disjoint polygons embedded in $\mathbb{R}^3$, where each edge may be shared by at most two polygons. Polyhedral surfaces have been long studied in computational geometry, and have well-established applications in for instance computer graphics [11] and geographical information science [8].

Inspired by those applications, classic work in this area often focuses on restricted cases, such as surfaces of (genus 0) polyhedra [3, 23], or x, y-monotone surfaces known as *polyhedral terrains* [7]. Such surfaces are, in a sense, 2-dimensional. One elegant way to capture this



"essentially 2-dimensional behaviour" is to look at the adjacency graph (see below for a precise definition) of the surface: in both cases described above, this graph is *planar*. In fact, by Steinitz's Theorem the adjacency graphs of surfaces of convex polyhedra are exactly the 3-connected planar graphs [35]. If we allow the surface of a polyhedron to have a *boundary*, then every planar graph has a representation as such a polyhedral surface [12].

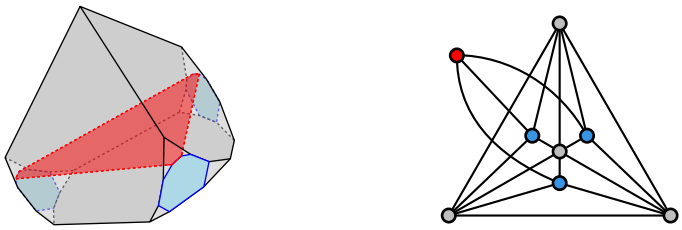
Recently, applications in computational topology have intensified the study of polyhedral surfaces of non-trivial topology. In sharp contrast to the simpler case above, where the classification is completely understood, little is known about the class of adjacency graphs that describe general polyhedral surfaces. In this paper we investigate this graph class.

Our model. A polyhedral surface $\mathcal{S} = \{S_1, \ldots, S_n\}$ is a set of n closed polygons embedded in $\mathbb{R}^3$ such that, for all pairwise distinct indices $i, j, k \in \{1, 2, \ldots, n\}$:

- S_i and S_j are interior-disjoint (w.r.t. the 2D relative interior of the objects);
- if $S_i \cap S_j \neq \emptyset$, then $S_i \cap S_j$ is either a single corner or a complete *side* of both S_i and S_j;
- if $S_i \cap S_j \cap S_k \neq \emptyset$ then it is a single corner (i.e., a side is shared by at most two polygons).

To avoid confusion with the corresponding graph elements, we consistently refer to polygon vertices as *corners* and to polygon edges as *sides*.

The *adjacency graph* of a polyhedral surface $\mathcal{S}$, denoted as $\mathcal{G}(\mathcal{S})$, is the graph whose vertices correspond to the polygons of $\mathcal{S}$ and which has an edge between two vertices if and only if the corresponding polygons of $\mathcal{S}$ share a side. Note that a corner–corner contact is allowed in our model but does not induce an edge in the adjacency graph. Further observe that the adjacency graph does not uniquely determine the topology of the surface. Fig. 1 shows an example of a polyhedral surface and its adjacency graph. We say that a polyhedral surface $\mathcal{S}$ *realizes* a graph G if $\mathcal{G}(\mathcal{S})$ is isomorphic to G. In this case, we write $\mathcal{G}(\mathcal{S}) \simeq G$.



(a) A surface $\mathcal{S}$. **(b)** The graph $\mathcal{G}(\mathcal{S})$.

Figure 1 A convex-polyhedral surface $\mathcal{S}$ and its nonplanar 3-degenerate adjacency graph $\mathcal{G}(\mathcal{S})$.

If every polygon of a polyhedral surface $\mathcal{S}$ is strictly convex, we call $\mathcal{S}$ a *convex-polyhedral* surface. Our paper focuses on convex-polyhedral surfaces; refer to Fig. 2 for an example of a general (nonconvex) polyhedral surface. We emphasize that we do not require that every polygon side has to be shared with another polygon.
Our work relates to two lines of research: Steinitz-type problems and contact representations.

Steinitz-type problems. Steinitz's Theorem gives the positive answer to the *realizability problem* for convex polyhedra. This result is typically stated in terms of the realizabilty of a graph as the 1-skeleton of a convex polyhedron. Our perspective comes from the dual point of view, describing the adjacencies of the faces instead of the adjacencies of the vertices.

Steinitz's Theorem settles the problem raised in this paper for surfaces that are homeomorphic to a sphere. A slightly stronger version of Steinitz's Theorem by Grünbaum and Barnette [5] states that every planar 3-connected graph can be realized as the 1-skeleton

of a convex polyhedron with the prescribed shape of one face. Consequently, also in our model we can prescribe the shape of one polygon if the adjacency graph of the surface is planar. For other classes of polyhedra only very few partial results for their graph-theoretic characterizations are known [13, 22]. No generalization for Steinitz's Theorem for surfaces of higher genus is known, and therefore there are also no results for the dual perspective. In higher dimensions, Richter-Gebert's Universality Theorem implies that the realizability problem for abstract 4-polytopes is $\exists\mathbb{R}$-complete [31].

The algorithmic problem of determining whether a given k-dimensional simplicial complex embeds in $\mathbb{R}^d$ is an active field of research [6, 17, 28, 30, 33, 34]. There exist at least three interesting notions of embeddability: linear, piecewise linear, and topological embeddability, which usually are not the same [28]. The case $(k, d) = (1, 2)$, however, corresponds to testing graph planarity, and thus, all three notions coincide, and the problem lies in P.

Contact representations. A realization of a graph as a polyhedral surface can be viewed as a *contact representation* of this graph with polygons in $\mathbb{R}^3$, where a contact between two polygons is realized by sharing an entire polygon side, and each side is shared by at most two polygons. In a general contact representation of a graph, the vertices are represented by interior-disjoint geometric objects, where two objects touch if and only if the corresponding vertices are adjacent. In concrete settings, the object type (disks, lines, polygons, etc.), the type of contact, and the embedding space is specified. Numerous results concerning which graphs admit a contact representation of some type are known; we review some of them.

The well-known Andreev–Koebe–Thurston circle packing theorem [2, 27] states that every planar graph admits a contact representation by touching disks in $\mathbb{R}^2$. A less known but impactful generalization by Schramm [32, Theorem 8.3] guarantees that every triangulation (i.e., maximal planar graph) has a contact representation in $\mathbb{R}^2$ where every inner vertex corresponds to a homothetic copy of a prescribed smooth convex set; the three outer vertices correspond to prescribed smooth arcs whose union is a simple closed curve. If the prototypes and the curve are polygonal, i.e., are not smooth, then there still exists a contact representation, however, the sets representing inner vertices may degenerate to points, which may lead to extra contacts. As observed by Gonçalves et al. [19], Schramm's result implies that every subgraph of a 4-connected triangulation has a contact representation with aligned equilateral triangles and similarly, every inner triangulation of a 4-gon without separating 3- and 4-cycles has a hole-free contact representation with squares [15].

While for the afore-mentioned existence results there are only iterative procedures that compute a series of representations converging to the desired one, there also exist a variety of shapes for which contact representations can be computed efficiently. Allowing for sides of one polygon to be *contained* in the side of adjacent polygons, Duncan et al. [12] showed that, in this model, every planar graph can be realized by hexagons in the plane and that hexagons are sometimes necessary. Assuming side–corner contacts, de Fraysseix et al. [10] showed that every plane graph has a triangle contact representation and how to compute one. Gansner et al. [18] presented linear-time algorithms for triangle side-contact representations for outerplanar graphs, square grid graphs, and hexagonal grid graphs. Kobourov et al. [26] showed that every 3-connected cubic planar graph admits a triangle side-contact representation whose triangles form a tiling of a triangle. For a survey of planar graphs that can be represented by dissections of a rectangle into rectangles, we refer to Felsner [15]. Alam et al. [1] presented a linear-time algorithm for hole-free contact representations of triangulations where each vertex is represented by a 10-sided rectilinear polygon of prescribed area.

Representations with one-dimensional objects in $\mathbb{R}^2$ have also been studied. While every plane bipartite graph has a contact representation with horizontal and vertical segments [9], recognizing segment contact graphs is an NP-complete problem even for planar graphs [20]. Hliněný showed that recognizing curve contact graphs, where no four curves meet in one point, is NP-complete for planar graphs and is solvable in polynomial time for planar triangulations.

Less is known about contact representations in higher dimensions. Every graph is the contact graph of interior-disjoint convex polytopes in $\mathbb{R}^3$ where contacts are shared 2-dimensional facets [37]. Hliněný and Kratochvíl [21] proved that the recognition of unit-ball contact graphs in $\mathbb{R}^d$ is NP-hard for $d = 3, 4$, and 8. Felsner and Francis [16] showed that every planar graph has a contact representation with axis-parallel cubes in $\mathbb{R}^3$. For proper side contacts, Kleist and Rahman [25] proved that every subgraph of an Archimedean grid can be represented with unit cubes, and every subgraph of a d-dimensional grid can be represented with d-cubes. Evans et al. [14] showed that every graph has a contact representation where vertices are represented by convex polygons in $\mathbb{R}^3$ and edges by shared corners of polygons, and gave polynomial-volume representations for bipartite, 1-planar, and cubic graphs.

Contribution and organization. We show that for every graph G there exists a polyhedral surface $\mathcal{S}$ such that G is the adjacency graph of $\mathcal{S}$; see Section 2. For convex-polyhedral surfaces, the situation is more intricate; see Section 3. Every planar graph can be realized by a *flat* convex-polyhedral surface (Proposition 3), i.e., a convex-polyhedral surface in $\mathbb{R}^2$. Some nonplanar graphs cannot be realized by convex-polyhedral surfaces in $\mathbb{R}^3$; in particular this holds for all supergraphs of K_5 (Proposition 5), of $K_{5,81}$ (Theorem 9), and of all nonplanar 3-trees (Theorem 14). Nevertheless, many nonplanar graphs, including $K_{4,4}$ and $K_{3,5}$, have such a realization (Propositions 7 and 8). We remark that all our positive results hold for subgraphs and subdivisions as well (Proposition 2). Similarly, our negative results carry over to supergraphs. For some proofs and additional figures, see the full version of this article [4].

Our results have implications on the maximum density of adjacency graphs of convex-polyhedral surfaces; see Section 4. While the non-realizability of $K_{5,81}$ implies that the number of edges of any realizable n-vertex graph is upperbounded by $\mathcal{O}(n^{9/5})$ edges, the realizability of hypercubes (Section 3.4) implies that it is in $\Omega(n \log n)$. Hence these graphs can be considerably denser than planar graphs, but not arbitrarily dense.

2 General Polyhedral Surfaces

We start with a positive result.

▶ **Proposition 1.** *For every graph G, there exists a polyhedral surface $\mathcal{S}$ such that $\mathcal{G}(\mathcal{S}) \simeq G$.*

Proof. We start our construction with $n = |V(G)|$ interior-disjoint rectangles such that there is a line segment s that acts as a common side of all these rectangles. We then cut away parts of each rectangle thereby turning it into a comb-shaped polygon as illustrated in Fig. 2. These polygons represent the vertices of G. For each pair (P, P') of polygons that are adjacent in G, there is a subsegment $s_{PP'}$ of s such that $s_{PP'}$ is a side of both P and P' that is disjoint from the remaining polygons. In particular, every polygon side is adjacent to at most two polygons. The result is a polyhedral surface whose adjacency graph is G. ◀

If we additionally insist that each polygon shares *all* of its sides with other polygons, the polyhedral surface describe a closed volume. In this model, K_7 can be realized as the Szilassi polyhedron; see Fig. 3. The tetrahedron and the Szilassi polyhedron are the only two known polyhedra in which each face shares a side with every other face [36]. Which other (complete) graphs can be realized in this way remains an open problem.

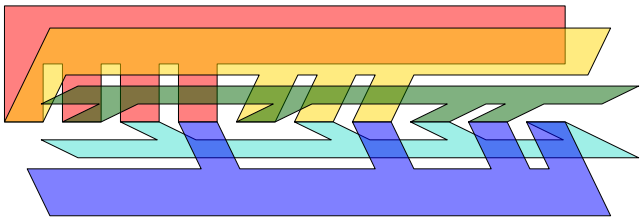


Figure 2 A realization of K_5 by arbitrary polygons with side contacts in $\mathbb{R}^3$.

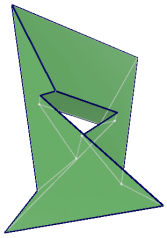


Figure 3 The Szilassi polyhedron realizes K_7 [36].

3 Convex-Polyhedral Surfaces

In this section we investigate which graphs can be realized by *convex*-polyhedral surfaces. First of all, it is always possible to represent a subgraph or a subdivision of an adjacency graph with slight modifications of the corresponding surface. While *trimming* the polygons allows to represent subgraphs, subdivision can be obtained by trimming and inserting chains of polygons. Consequently, we obtain the following result, which we prove formally in the full version of this article [4].

▶ **Proposition 2.** *The set of adjacency graphs of convex-polyhedral surfaces in $\mathbb{R}^3$ is closed under taking subgraphs and subdivisions.*

The existence of a flat surface with the correct adjacencies follows from the Andreev–Koebe–Thurston circle packing theorem; we include a direct proof in the full version [4].

▶ **Proposition 3.** *For every planar graph G, there exists a flat convex-polyhedral surface $\mathcal{S}$ such that $\mathcal{G}(\mathcal{S}) \simeq G$. Moreover, such a surface can be computed in linear time.*

So for planar graphs, corner and side contacts behave similarly. For nonplanar graphs (for which the third dimension is essential), the situation is different. Here, side contacts are more restrictive.

3.1 Complete Graphs

We introduce the following notation. In a polyhedral surface $\mathcal{S}$ with adjacency graph G, we denote by P_v the polygon in $\mathcal{S}$ that represents vertex v of G.

▶ **Lemma 4.** *Let $\mathcal{S}$ be a convex-polyhedral surface in $\mathbb{R}^3$ with adjacency graph G. If G contains a triangle uvw, polygons P_v and P_w lie in the same closed halfspace w.r.t. P_u.*

Proof. Due to their convexity, each of P_v and P_w lie entirely in one of the closed halfspaces with respect to the supporting plane of P_u. Moreover, one of the halfspaces contains both P_v and P_w; otherwise they cannot share a side and the edge vw would not be represented. ◀

A graph H is *subisomorphic* to a graph G if G contains a subgraph G' with $H \simeq G'$.

▶ **Proposition 5.** *There exists no convex-polyhedral surface $\mathcal{S}$ in $\mathbb{R}^3$ such that K_5 is subisomorphic to $\mathcal{G}(\mathcal{S})$.*

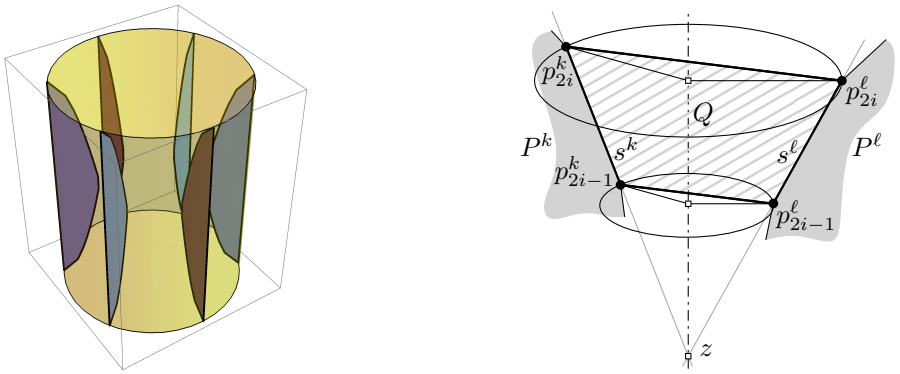
Proof. Suppose that there is a convex-polyhedral surface $\mathcal{S}$ with $\mathcal{G}(\mathcal{S}) \simeq K_5$. By Lemma 4 and the fact that all vertex triples form a triangle, the surface $\mathcal{S}$ lies in one closed halfspace of the supporting plane of every polygon P of $\mathcal{S}$. In other words, $\mathcal{S}$ is a subcomplex of a (weakly) convex polyhedron, whose adjacency graph must be planar. This yields a contradiction to the nonplanarity of K_5. Together with Proposition 2 this implies the claim. ◀

Evans et al. [14] showed that every bipartite graph has a contact representation by touching polygons on a polynomial-size integer grid in $\mathbb{R}^3$ for the case of corner contacts. As we have seen before, side contacts are less flexible. In particular, in Theorem 9 we show that $K_{5,81}$ cannot be represented. On the positive side, we show in the following that those bipartite graphs that come from subdividing edges of arbitrary graphs can be realized. In our construction, we place the polygons in a cylindrical fashion, which is reminiscent to the realizations created by Evans et al. However, due to the more restrictive nature of side contacts, the details of the two approaches are necessarily quite different.

▶ **Theorem 6.** *Let G be any graph, and let G' be the subdivision of G in which every edge is subdivided with (at least) one vertex. Then there exists a convex-polyhedral surface $\mathcal{S}$ in $\mathbb{R}^3$ such that $\mathcal{G}(\mathcal{S}) \simeq G'$.*

Proof. Let $V(G) = \{v_1, \ldots, v_n\}$, let $E(G) = \{e_1, \ldots, e_m\}$, and let P be a strictly convex polygon with corners $p_1, \ldots, p_{2m}$ in the plane. We assume that $m \geq 2$, that p_1 and p_{2m} lie on the x-axis, and that the rest of the polygon is a convex chain that projects vertically onto the line segment $\overline{p_1 p_{2m}}$, which we call the *long side* of P. We call the other sides *short sides*. We choose P such that no short side is parallel to the long side.

Let Z be a (say, unit-radius) cylinder centered at the z-axis. For each vertex v_i of G, we take a copy P^i of P and place it vertically in $\mathbb{R}^3$ such that its long side lies on the boundary of Z; see Fig. 4a. Each polygon P^i lies inside Z on a distinct halfplane that is bounded by the z-axis. Finally, all polygons are positioned at the same height, implying that for any $j \in \{1, \ldots, 2m\}$, all copies of p_j lie on the same horizontal plane h_j and have the same distance to the z-axis.



(a) polygons $P^1, \ldots, P^n$. **(b)** quadrilateral Q spanned by s^k and s^ℓ.

Figure 4 Illustration for the proof of Theorem 6.

Let $i \in \{1, \ldots, m\}$. Then the side $s = p_{2i-1}p_{2i}$ is a short side of P. For $k = 1, 2, \ldots, n$, we denote by s^k and p^k_i the copies of s and p_i in P^k, respectively. We claim that, for $1 \leq k < \ell \leq n$, the sides s^k and s^ℓ span a convex quadrilateral that does not intersect any P^j with $j \notin \{k, \ell\}$. To prove the claim, we argue as follows; see Fig. 4b.

By the placement of P^k and P^ℓ inside Z, the supporting lines of s^k and s^ℓ intersect at a point z on the z-axis, implying that s^k and s^ℓ are coplanar. Moreover, p^k_{2i-1} and p^ℓ_{2i-1} are at the same distance from z, and the same holds for p^k_{2i} and p^ℓ_{2i}. Hence the triangle spanned by z, p^k_{2i-1}, and p^ℓ_{2i-1} is similar to the triangle spanned by z, p^k_{2i}, and p^ℓ_{2i}, implying that $p^k_{2i-1}p^\ell_{2i-1}$ and $p^k_{2i}p^\ell_{2i}$ are parallel and hence span a convex quadrilateral Q (actually a trapezoid). Finally, no polygon P^j with $j \notin \{k, \ell\}$ can intersect Q as any point in the interior of Q lies closer to the z-axis than any point of P^j at the same z-coordinate, which proves the claim.

We use Q as polygon for the subdivision vertex of the edge e_i of G (in case e_i was subdivided multiple times, we dissect Q accordingly). Let v_a and v_b be the endpoints of e_i. By our claim, Q that does not intersect any P^j with $j \notin \{a, b\}$. The quadrilateral Q lies in the horizontal slice of Z bounded by the horizontal planes h_{2i-1} and h_{2i}. Since any two such slices are vertically separated and hence disjoint, the m quadrilaterals together with the n copies of P constitute a valid representation of G'. ◀

Note that the combination of Proposition 5 and Theorem 6 rules out any Kuratowski-type characterization for adjacency graphs of convex-polyhedral surfaces. This graph class contains a subdivision of K_5, but it does not contain K_5; hence it is not minor-closed.

3.2 Complete Bipartite Graphs

▶ **Proposition 7.** *There exists a convex-polyhedral surface $\mathcal{S}$ such that $\mathcal{G}(\mathcal{S}) \simeq K_{4,4}$.*

Proof sketch. Start with a rectangular box in $\mathbb{R}^3$ and stab it with two rectangles that intersect each other in the center of the box as indicated in Fig. 5 (left). We can now draw polygons on these eight rectangles such that each of the four vertical rectangles (representing the four vertices of one side of the bipartition of $K_{4,4}$) contains a polygon that has a side contact with a polygon on each of the four horizontal or slanted rectangles (representing the other side of the bipartition of $K_{4,4}$). To remove the intersection of the (polygons drawn on the) two slanted rectangles, we shift one corner of the original box; see Fig. 5 (center and right). A description of the resulting polygons via their coordinates and more figures can be found in the full version [4]. Note that, in the resulting representation, there are two additional side contacts between pairs of polygons that are colored blue in Fig. 5 (right). By Proposition 2, these contacts can be removed. ◀

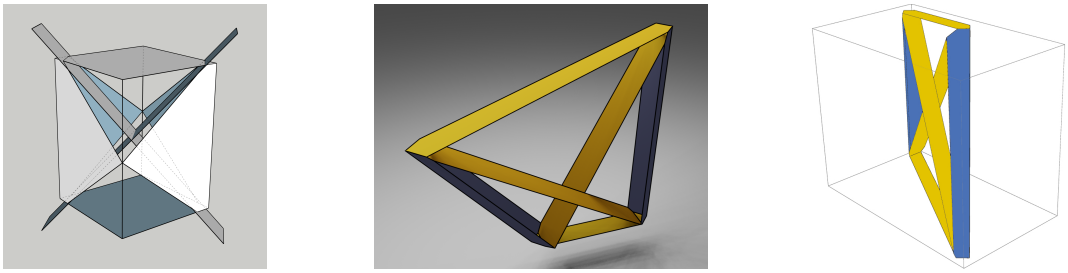


Figure 5 Construction of a convex-polyhedral surface $\mathcal{S}$ with $\mathcal{G}(\mathcal{S}) \simeq K_{4,4}$. The 2-coloring of the polygons in the central figure reflects the bipartition of $K_{4,4}$.

▶ **Proposition 8.** *There exists a convex-polyhedral surface $\mathcal{S}$ such that $\mathcal{G}(\mathcal{S}) \simeq K_{3,5}$.*

Proof. We call the vertices of the smaller bipartition class the gray vertices, and their polygons gray polygons. For the other class we pick a distinct color for every vertex and use the same naming-by-color convention. We start our construction with a triangular prism in which the quadrilateral faces q_1, q_2, q_3 are rectangles of the same size. Each of the faces q_i will contain one gray polygon. All colorful polygons lie inside the prism. We call the lines resulting from the intersection of the supporting planes with the prism the *colorful supporting lines*. Unfolding the faces q_1, q_2, and q_3 yields Fig. 6, which shows the gray polygons and the colorful supporting lines. Note that the vertices of the gray polygons in the figure are actually very small edges that have the slope of the colorful supporting line there are placed on. The colorful polygons are now already determined.

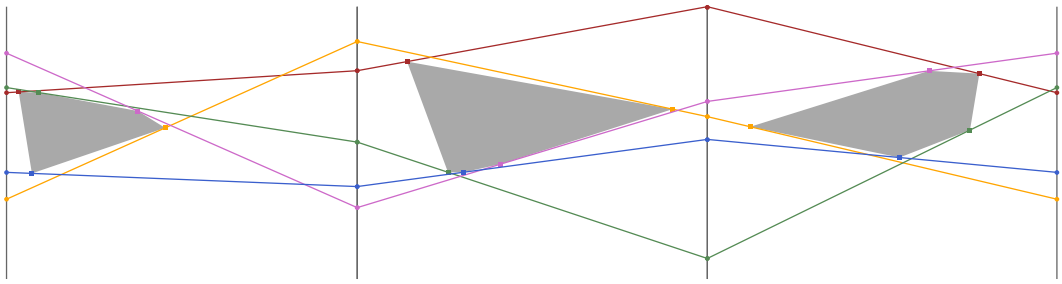


Figure 6 Constructing a convex-polyhedral surface whose adjacency graph is isomorphic to $K_{3,5}$. The prism boundary is unfolded into the plane.

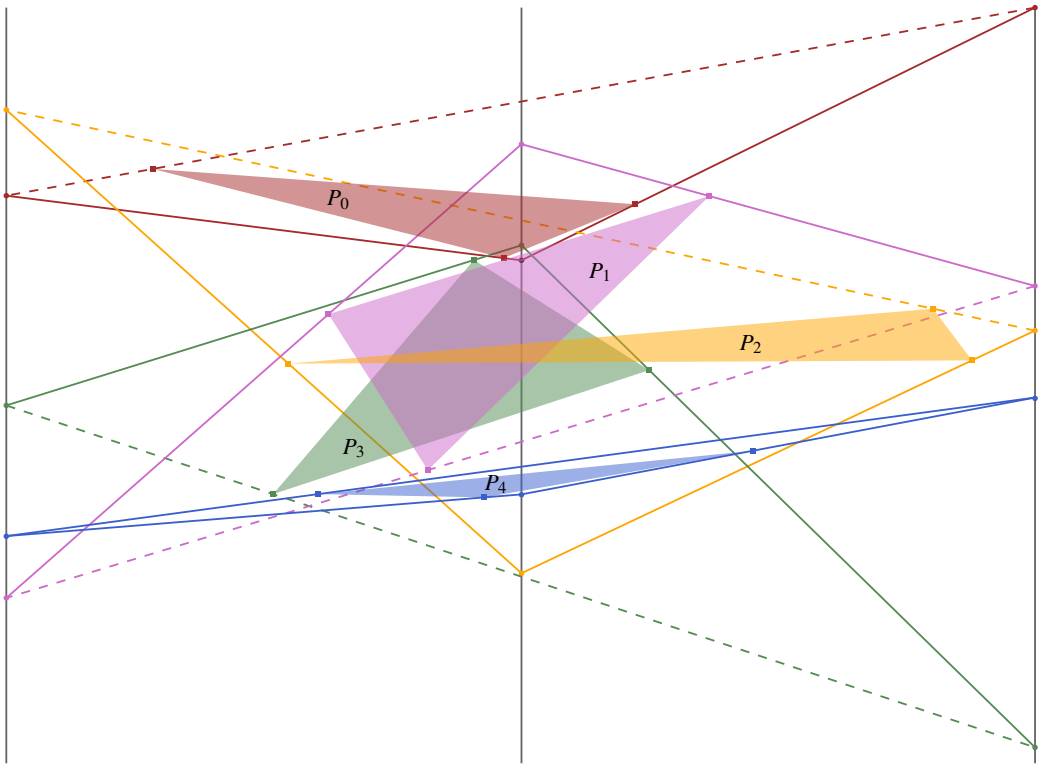


Figure 7 Front view of the prism containing a realization of $K_{3,5}$. Lines on the back are dashed.

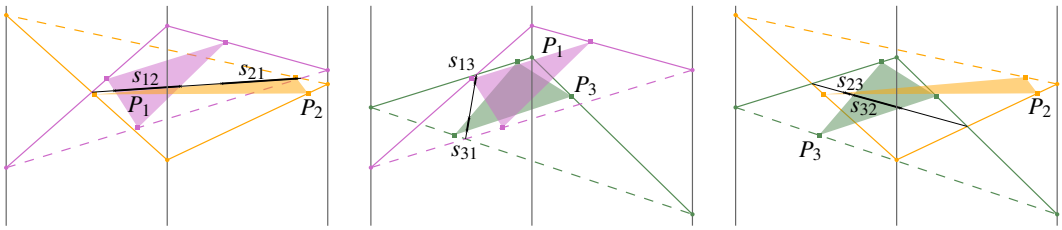


Figure 8 Certificates for the disjointness of colorful polygons P_1, P_2, and P_3. The supporting planes intersect in the thin black lines; thicker segments indicate where the lines intersect polygons.

We are left with checking that the colorful polygons are disjoint. Fig. 7 shows the prism in a view from the side where we dashed all objects on the hidden prism face. The brown polygon P_0 and the blue polygon P_4 avoid all other colorful polygons in this projection and thus they avoid all other polygons in $\mathbb{R}^3$, too.

For the pink polygon P_1, the orange polygon P_2 and the green polygon P_3, we proceed as follows to prove disjointness. Pick two of the polygons and name them P_i and P_j. The line ℓ_{ij} of intersection of the supporting planes of P_i and P_j is determined by the two intersections of the corresponding colorful supporting lines. If the polygons intersect, they have to intersect on this line. Polygon P_i intersects ℓ_{ij} in a segment s_{ij}; polygon P_j intersects ℓ_{ij} in s_{ji}. Fig. 8 shows, however, that s_{ij} and s_{ji} do not overlap in any of the three cases. ◀

In contrast to Propositions 7 and 8, we can show that not every complete bipartite graph can be realized as a convex-polyhedral surface in $\mathbb{R}^3$.

▶ **Theorem 9.** *There exists no convex-polyhedral surface $\mathcal{S}$ in $\mathbb{R}^3$ such that $K_{5,81}$ is subisomorphic to $\mathcal{G}(\mathcal{S})$.*

To prove the theorem we start with some observations about realizing complete bipartite graphs. We will consider a set R of red polygons, and a set B of blue polygons, so that each red–blue pair must have a side contact. For all $p \in R \cup B$ we denote by $p^=$ the supporting plane of p, by p^- the closed half-space left of $p^=$, and by p^+ the closed half-space right of $p^=$ (orientations can be chosen arbitrarily). We start with a simpler setting where we have an additional constraint. We call B *one-sided w.r.t.* R if all red polygons must be on the same half-space of each blue polygon, i.e., $\forall b \in B\colon ((\forall r \in R\colon r \subseteq b^-) \vee (\forall r \in R\colon r \subseteq b^+))$.

▶ **Lemma 10.** *Let R and B be two sets of convex polygons in $\mathbb{R}^3$ realizing $K_{|R|,|B|}$. If $|R| = 3$ and B is one-sided w.r.t. R, then $|B| \leq 8$.*

Proof. Let $R = \{r_1, r_2, r_3\}$ and let $\mathcal{A}$ be the arrangement of the supporting planes of R. Consider a polygon $b \in B$. For every polygon $r_i \in R$, since b is convex and shares a side with r_i, b is contained in r_i^- or r_i^+. Thus, b is contained in a cell of $\mathcal{A}$. Let $r_* = r_1^= \cap r_2^= \cap r_3^=$ be the intersection of the supporting planes of R. We may assume that no two supporting planes of R coincide; otherwise, by strict convexity, all polygons (in B and finally in $B \cup R$) must lie in the same plane and the non-planarity of $K_{3,3}$ implies that $|B| \leq 2$. Further, if r_* is a line, then every b lies in a cell C, in which one of the red polygons is only present as a subset of r_*. It is not possible to add b such that it has a common segment with r_* and each of the bounding planes of C. Consequently, r_* is either a point or the empty set. We consider two cases: either (1) $r_* = \emptyset$ or no polygon in R contains the point r_*, or (2) one polygon in R does contain the point r_*.

Case (1). Either $r_* = \emptyset$ or no polygon in R contains the point r_*. We claim that at most four cells of $\mathcal{A}$ are incident to all polygons of R, and at most two polygons of B exist per cell.

If $r_* = \emptyset$, $\mathcal{A}$ contains two parallel planes or forms an infinite prism. It is easy to check that $\mathcal{A}$ has at most four cells that are incident to three planes. If $r_* \neq \emptyset$, there are eight cells, called *octants*, of space of the form $Q^{a_0 b_0 c_0} = r_1^{a_0} \cap r_2^{b_0} \cap r_3^{c_0}$, $a_0, b_0, c_0 \in \{+, -\}$. If the point r_* is disjoint from all r_i, then each r_i intersects at most six octants: r_1 rules out the two octants $Q^{\pm b_1 c_1}$, r_2 "rules out" the two octants $Q^{a_2 \pm c_2}$, and r_3 rules out the two octants $Q^{a_3 b_3 \pm}$ for some $a_2, a_3, b_1, b_3, c_1, c_2 \in \{+, -\}$. The maximal number of octants that "remain" is four. For example one could pick $a_2 = a_3 = b_1 = b_3 = c_1 = c_2 = +$. By this choice, $Q^{+--}, Q^{-+-}, Q^{--+}$, and Q^{---} remain. All other choices (a finite set to check) will have at most four octants touching all three polygons.

Now consider one such cell C incident to all r_i. For the argument within this cell, we can truncate r_i to C. Since each r_i lies in a boundary plane of C and we are looking for a polygon b that has all of R on one side, the four polygons r_1, r_2, r_3, b must be in convex position. Assume that we have three such polygons b_1, b_2, and b_3. The cell C has three (unbounded) boundary faces f_1, f_2, and f_3 such that r_j lies in f_j. Let $\ell_{i,j}$ be the segment that is the intersection of the plane $b_i^=$ and the polygon f_j, and let t_i be the triangle formed by $\ell_{i,1}$, $\ell_{i,2}$, $\ell_{i,3}$. Two of these triangles intersect either in two points or they are disjoint. On no face f_j, all three $\ell_{1,j}$, $\ell_{2,j}$, $\ell_{3,j}$ can be disjoint since then the polygon belonging to the "middle segment" intersects the interior of r_j. On the other hand, every pair t_i, $t_{i'}$ has one face f_j without intersection. Since we have three possible pairs, every face f_j contains two segments $\ell_{i,j}$ and $\ell_{i',j}$ that do not intersect. Furthermore, the pair i, i' will be different for every f_j. We call the corresponding parallel segments *upper* and *lower*, depending on whether r_j lies below or above the corresponding plane ($b_i^=$ or $b_{i'}^=$). Now we have a contradiction to B being one-sided with respect to R since there has to be one t_i that belongs to an upper segment on one face f_j and to a lower segment on another face $f_{j'}$. In other words, not all polygons in R lie on the same side of $b_i^=$. Thus, at most two polygons of B lie in C, one corresponding to upper segments and one corresponding to lower segments. This yields at most $2 \cdot 4 = 8$ polygons in B in Case (1).

Case (2). One red polygon contains r_*, w.l.o.g. this is r_1. We claim that at most five cells of $\mathcal{A}$ are incident to all polygons of R, and at most one polygon of B exist per cell.

Similar as in Case (1), now r_2 and r_3 both intersect at most 6 octants, and there are at most five octants that intersect every polygon in R. Consider again some octant Q. Since r_1 contains r_*, now there may be only one polygon $b \in B$ that is contained in Q and has all polygons in R on the same side of $b^=$ (the lower segments in Case (1) are now forbidden). Thus, there are at most $1 \cdot 5 = 5$ polygons in B in Case (2). ◀

With the help of Lemma 10 we can now prove Theorem 9.

Proof of Theorem 9. Assume that $K_{5,81}$ can be realized, and let R be a set of five red polygons. Since every $b \in B$ is adjacent to all polygons in R, b partitions R into two sets: those in b^- and those in b^+. At least one of these subsets must have at least three elements. Arbitrarily charge b to such a set of three polygons. By Lemma 10, each set of three red polygons can be charged at most eight times. There are $\binom{5}{3} = 10$ sets of three red polygons. Therefore, there can be at most $8 \cdot 10 = 80$ blue polygons; a contradiction. Together with Proposition 2 this implies the claim. ◀

3.3 3-Trees

The graph class of 3-trees is recursively defined as follows: K_4 is a 3-tree. A graph obtained from a 3-tree G by adding a new vertex x with exactly three neighbors u, v, w that form a triangle in G is a 3-tree. We say x is *stacked* on the triangle uvw. It follows that for each 3-tree there exists a (not necessarily unique) *construction sequence* of 3-trees $G_4, G_5, \ldots, G_n$ such that $G_4 \simeq K_4$, $G_n = G$, and where for $i = 4, 5, \ldots, n-1$ the graph G_{i+1} is obtained from G_i by stacking a vertex v_{i+1} on some triangle of G_i.

By Proposition 3, for every planar 3-tree G there is a polyhedral surface $\mathcal{S}$ (even in $\mathbb{R}^2$) with $\mathcal{G}(\mathcal{S}) \simeq G$. On the other hand, we can show that no nonplanar 3-tree has such a realization in $\mathbb{R}^3$. To this end, we observe that a 3-tree is nonplanar if and only if it contains the *triple-stacked triangle* as a subgraph. The triple-stacked triangle is the graph that consists of $K_{3,3}$ plus a cycle that connects the vertices of one part of the bipartition; see Fig. 9. We show that the triple-stacked triangle is not realizable.

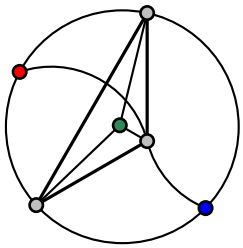


Figure 9 The unique minimal nonplanar 3-tree, which we call triple-stacked triangle.

▶ **Lemma 11.** *Let uvw be a separating triangle in a plane 3-tree $G = (V, E)$. Then there exist vertices $a, b \in V$ that belong to distinct sides of uvw in G such that both $\{a, u, v, w\}$ and $\{b, u, v, w\}$ induce a K_4 in G.*

Proof. Let $G_4, G_5, \ldots, G_n$ denote a construction sequence of $G = G_n$, and let k be the largest index in $\{4, 5, \ldots, n\}$ such that uvw is nonseparating in G_k. Since uvw is separating in G_{k+1}, it follows that the vertex $v_{k+1} = a$ is stacked on uvw (say, inside uvw) to obtain G_{k+1} and, hence, $\{a, u, v, w\}$ induce a K_4 in G_{k+1} and G.

It remains to argue about the existence of the vertex b in the exterior of uvw. If uvw is one of the triangles of the original $G_4 \simeq K_4$, there is nothing to show, so assume otherwise. Let j be the smallest index in $\{5, 6, \ldots, n\}$ such that uvw is contained in G_j. It follows that one of u, v, w, say u, is the vertex v_j that was stacked on some triangle xyz of G_{j-1} to obtain G_j. Without loss of generality, we may assume that $\{v, w\} = \{y, z\}$. It follows that $x = b$ forms a K_4 with u, v, w in G_j and G. ◀

▶ **Lemma 12.** *A 3-tree is nonplanar iff it contains the triple-stacked triangle as a subgraph.*

Proof. The triple-stacked triangle is nonplanar because it contains a $K_{3,3}$ (one part of the bipartition is formed by the gray vertices and the other by the colored vertices).

For the other direction, let G be a nonplanar 3-tree. Let $G_4, G_5, \ldots, G_n$ be a construction sequence of G. Let k be the smallest index in $\{4, 5, \ldots, n\}$ such that G_k is nonplanar. By 3-connectivity, the graph G_{k-1}, which is planar, has a unique combinatorial embedding. Therefore, we may consider G_{k-1} to be a plane graph. Let uvw be the triangle that the vertex v_k was stacked on to obtain G_k from G_{k-1}. Since G_k is nonplanar, the triangle uvw is a separating triangle of G_{k-1}. It follows by Lemma 11 that G_k (and, hence, G) contains the triple-stacked triangle. ◀

▶ **Lemma 13.** *There exists no convex-polyhedral surface $\mathcal{S}$ in $\mathbb{R}^3$ such that triple-stacked triangle is subisomorphic to $\mathcal{G}(\mathcal{S})$.*

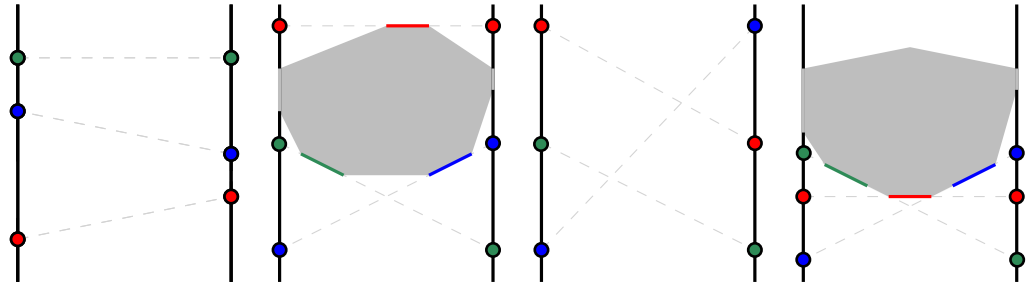
Proof. We refer to the vertices of the triple-stacked triangle as the three gray vertices and the three colored (red, green, and blue) vertices; see also Fig. 9. Given the correspondence between vertices and polygons (and their supporting planes), we also refer to the polygons (and the supporting planes) as gray and colored.

Assume that the triple-stacked triangle can be realized. Consider the arrangement of the gray supporting planes. By strict convexity, it follows that if a pair of gray polygons has the same supporting plane, then all their common neighbors lie in the same plane. This implies that all supporting planes coincide – a contradiction to the non-planarity of the triple-stacked triangle. Consequently, the gray supporting planes are pairwise distinct. (Likewise, it holds that no colored and gray supporting plane coincide.)

We now argue that all colored polygons are contained in the same closed cell of the gray arrangement. To see this, fix one gray polygon and observe, by Lemma 4, that all polygons are contained in the same closed half space with respect to its supporting plane.

Note that gray plane arrangement has one of the following two combinatorics: either the three planes have a common point of intersection (cone case) or not (prism-case). In the first case, the planes partition the space into eight cones, one of which contains all polygons; in the second case, the (unbounded) cell containing all polygons forms a (unbounded) prism. For a unified presentation, we transform any occurrence of the first case into the second case. To do so, we move the apex of the cone containing all polygons to the point at infinity by a projective transformation. This turns each face of the cone into a strip that is bounded by two of the extremal rays of the cone, which now has been deformed into a prism.

Consider one of the strips, which we call S. The strip S has to contain one of the gray polygons, which we call P_S. We know that P_S has at least five sides, one for each neighbor. Each of the two *bounding lines* contains a side to realize the adjacency to the other two gray polygons. We call the sides of P_S that realize the adjacencies to the remaining polygons red, green, and blue, in correspondence to the vertex colors. The supporting line of the red side intersects each bounding line of S. We add a red point at each of the intersections. For the blue and green sides we proceed analogously. By convexity of P_S, these points are distinct. This yields a permutation of red, green, blue (see Fig. 10) on each bounding line. The permutations on the boundary of two adjacent strips coincide because the supporting lines are clearly contained in the supporting planes.



(a) 0 inversions. **(b)** 1 inversion. **(c)** 2 inversions. **(d)** 3 inversions.

Figure 10 The permutations of the intersections with the supporting lines of the red, green, and blue edges as in the proof of Lemma 13. Figures (a) and (c) illustrate possible scenarios. Figures (b) and (d) show impossible scenarios because they do not contain cells of complexity 5.

Consider the line arrangement inside S given by the supporting lines of the red, green, and blue sides. Up to symmetry, Fig. 10 illustrates the different intersection patterns. To realize all contacts, the polygon P_S has to lie inside a cell incident to all five lines, namely the two bounding lines and the supporting lines. It is easy to observe that such a cell exists only if the permutation has exactly one or three inversions, see Figs. 10b and 10d. In particular, the number of inversions is odd.

Following the cyclic order of the bounding lines around the prism, we record three odd numbers of inversions in the permutations before coming back to the start. Since an odd number of inversions does not yield the identity, we obtain the desired contradiction. ◀

Together Lemmas 12 and 13 yield the following theorem.

▶ **Theorem 14.** *Let G be a 3-tree. There exists a convex-polyhedral surface $\mathcal{S}$ in $\mathbb{R}^3$ with $\mathcal{G}(\mathcal{S}) \simeq G$ if and only if G is planar.*

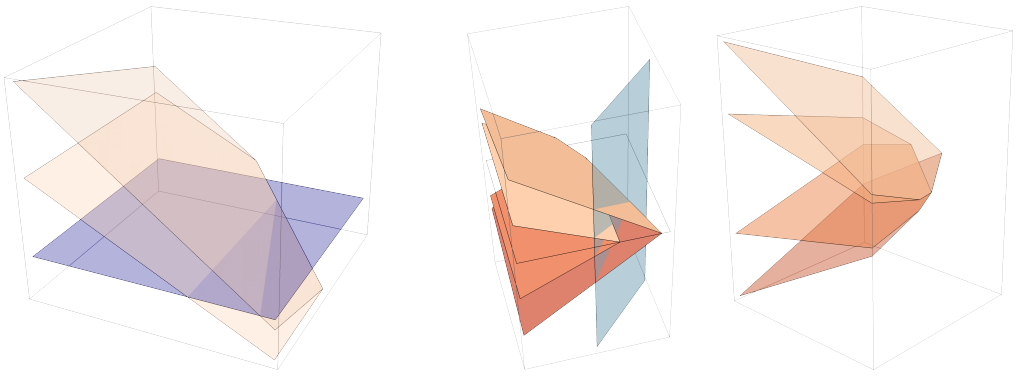
In contrast to Theorem 14, there are nonplanar 3-degenerate graphs that can be realized; see the example in Fig. 1.

3.4 Hypercubes

In a paper from 1983, McMullen, Schulz, and Wills construct a polyhedron for every integer $p \geq 4$ such that all faces are convex p-gons [29, Sect. 4]. In the following, we show and illustrate how their result proves the realizability of any hypercube.

▶ **Proposition 15** ([29]). *For every d-hypercube Q_d, $d \geq 0$, there exists a convex-polyhedral surface $\mathcal{S}$ in $\mathbb{R}^3$ with $\mathcal{G}(\mathcal{S}) \simeq Q_d$ and every polygon of $\mathcal{S}$ is a $(d+4)$-gon.*

The main building block in their construction is a polyhedral surface whose adjacency graph is a $(p-4)$-hypercube. In fact, we observed that the adjacency graph of the polyhedron they finally construct is the Cartesian product of Q_{p-4} and a cycle graph $C_n, n \geq 3$. For the first few steps of their inductive construction, see Fig. 11.



(a) Realization of Q_1 (red), xy-plane (blue). **(b)** Inductive step. **(c)** Realization of Q_2.

Figure 11 The inductive construction of McMullen et al. [29].

Recall that the d-hypercube has 2^d vertices. The base case for $d = 0$ is given by a single 4-gon, namely by the unit square. What follows is a series of inductive steps. In every step, the value of d increases by one and the number of polygons doubles. Before explaining the step, we state the invariants of the construction. After every step, all polygons have (almost) the same orthogonal projection into the xy-plane. Furthermore, this projection looks like the unit square in which we have replaced the upper right corner with a convex chain as shown in Fig. 12(a). The sides on the convex chain (with negative slopes) have already two incident polygons, the four other sides are currently incident to only one polygon. When projecting the polygons into the xy-plane only the convex chain edges differ.

We explain next how to execute the inductive step. Suppose that we have a polyhedral surface where every polygon is a $(d+4)$-gon fulfilling our invariant. We apply a shear along the z-axis and a vertical shift to the whole surface such that exactly the sides at $x = 1$ lie completely below the xy-plane, see also Fig. 11a. These transformations do not change the projections of the polygons to the xy-plane. We then cut the surface with the xy-plane. By this we slice away one of the sides in all polygons but also add a side that lies in the xy-plane; see Fig. 12(b). Each polygon now has a side that lies in the xy-plane and is disjoint from all other polygons. We now take a copy of the surface at hand and reflect it across the xy-plane. Every polygon of the original (unreflected) surfaces is now glued to its reflected copy via the common side in the xy-plane. With this step, we already have transformed the adjacency

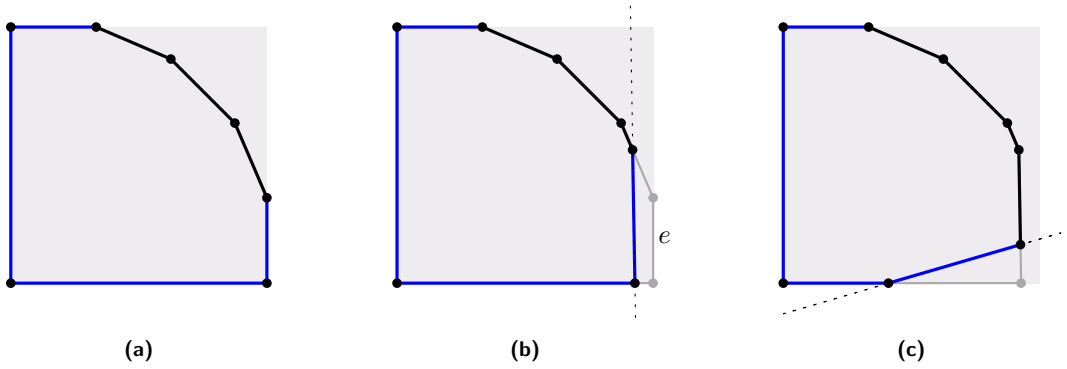


Figure 12 Projection of a polygon into the xy-plane in the construction of McMullen et al. The gray rectangle depicts the unit square. Edges incident to only one polygon are drawn in blue. (a) The start configuration for $d + 4 = 7$. (b) Cutting with the xy-plane after the shear that puts only e below the xy-plane (before glueing the reflected copy). (c) Slicing off a corner to get the initial situation for $d + 4 = 8$ modulo a projective transformation.

graph from a d-hypercube to a $(d + 1)$-hypercube. We only need to bring the surface back into the shape required by the invariant. To do so, we cut off a corner (see Fig. 12(c)) by slicing the whole construction with an appropriate plane, see also Fig. 11b. This turns all $(d + 4)$-gons into $(d + 5)$-gons. Finally, we apply a projective transformation to restore a required shape. Figures 11a–11c show spatial images of this construction.

4 Bounds on the Density

It is an intriguing question how dense adjacency graphs of convex-polyhedral surfaces can be. In this section, we use realizability and non-realizability results from the previous sections to derive asymptotic bounds on the maximum density of such graphs, which we phrase in terms of the relation between their number of vertices and edges.

Let $\mathcal{G}_n$ be the class of graphs on n vertices with a realization as a convex-polyhedral surface in $\mathbb{R}^3$. Further, let $e_{\max}(n) = \max_{G \in \mathcal{G}_n} |E(G)|$ be the maximum number of edges that a graph in $\mathcal{G}_n$ can have.

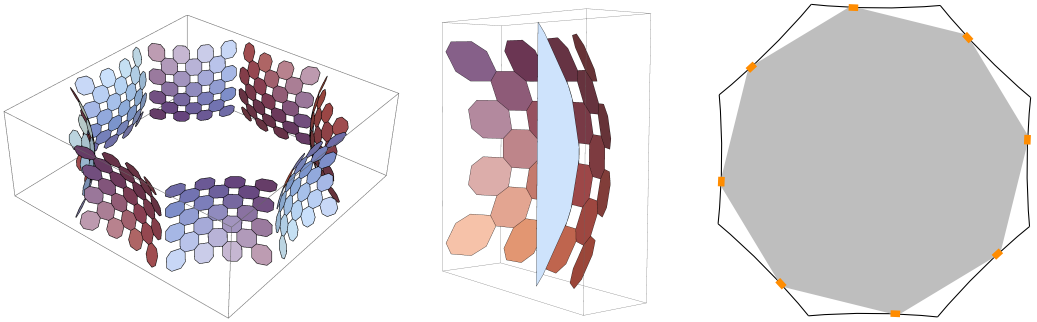
▶ **Corollary 16.** *$e_{\max}(n) \in \Omega(n \log n)$ and $e_{\max}(n) \in \mathcal{O}(n^{9/5})$.*

Proof. For the lower bound, note that by Proposition 15, every hypercube is the adjacency graph of a convex-polyhedral surface. As the d-dimensional hypercube has 2^d vertices and $2^d \cdot d/2$ edges, the bound follows.

For the upper bound, we use that, by Theorem 9, the adjacency graph of a convex-polyhedral surface cannot contain $K_{5,81}$ as a subgraph. It remains to apply the Kővari–Sós–Turán Theorem [24], which states that an n-vertex graph that has no $K_{s,t}$ as a subgraph can have at most $\mathcal{O}(n^{2-1/s})$ edges. ◀

Before being aware of the result of McMullen et al. [29], we constructed a family of surfaces with (large, but) constant average degree; see the full version [4]. Our construction is not recursive and therefore easier to understand and visualize; see Fig. 13. Note that some polygons in our construction have polynomial degree.

▶ **Proposition 17.** *There is an unbounded family of convex-polyhedral surfaces in $\mathbb{R}^3$ whose adjacency graphs have average vertex degree $12 - o(1)$.*



(a) placement of octagon grids. **(b)** a vertical polygon. **(c)** a horizontal polygon from above.

Figure 13 An family of convex-polyhedral surfaces whose adjacency graphs have average vertex degree $12 - o(1)$. The vertical polygons are attached to the "outside" of the grids; the horizontal polygons touch each grid along a single polygon side.

5 Conclusion and Open Problems

In this paper, we have studied the question which graphs can be realized as adjacency graphs of (convex-)polyhedral surfaces. In Corollary 16, we bound the maximum number $e_{\max}(n)$ of edges in realizable graphs on n vertices by $\Omega(n \log n)$ and $\mathcal{O}(n^{9/5})$. It would be interesting to improve upon these bounds. We conjecture that realizability is NP-hard to decide.

References

1 Md. Jawaherul Alam, Therese C. Biedl, Stefan Felsner, Andreas Gerasch, Michael Kaufmann, and Stephen G. Kobourov. Linear-time algorithms for hole-free rectilinear proportional contact graph representations. *Algorithmica*, 67(1):3–22, 2013. doi:10.1007/s00453-013-9764-5.

2 E. M. Andreev. Convex polyhedra in Lobačevskiĭ spaces. *Mat. Sb. (N.S.)*, 81 (123)(3):445–478, 1970. doi:10.1070/SM1970v010n03ABEH001677.

3 Boris Aronov, Marc J. van Kreveld, René van Oostrum, and Kasturi R. Varadarajan. Facility location on a polyhedral surface. *Discret. Comput. Geom.*, 30(3):357–372, 2003. doi:10.1007/s00454-003-2769-0.

4 Elena Arseneva, Linda Kleist, Boris Klemz, Maarten Löffler, André Schulz, Birgit Vogtenhuber, and Alexander Wolff. Adjacency graphs of polyhedral surfaces. ArXiv report, 2021. arXiv:2103.09803.

5 David W. Barnette and Branko Grünbaum. On Steinitz's theorem concerning convex 3-polytopes and on some properties of planar graphs. In G. Chartrand and S. F. Kapoor, editors, *The Many Facets of Graph Theory*, pages 27–40. Springer Berlin Heidelberg, 1969.

6 Martin Čadek, Marek Krčál, and Lukáš Vokřínek. Algorithmic solvability of the lifting-extension problem. *Discrete Comput. Geom.*, 57(4):915–965, 2017. doi:10.1007/s00454-016-9855-6.

7 Richard Cole and Micha Sharir. Visibility problems for polyhedral terrains. *J. Symb. Comput.*, 7(1):11–30, 1989. doi:10.1016/S0747-7171(89)80003-3.

8 Leila de Floriani, Paola Magillo, and Enrico Puppo. Applications of computational geometry in geographic information systems. In J.R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, chapter 7, pages 333–388. Elsevier, Amsterdam, 1997.

9 Hubert de Fraysseix, Patrice Ossona de Mendez, and János Pach. Representation of planar graphs by segments. *Intuitive Geometry*, 63:109–117, 1991. URL: https://infoscience.epfl.ch/record/129343/files/segments.pdf.

10 Hubert de Fraysseix, Patrice Ossona de Mendez, and Pierre Rosenstiehl. On triangle contact graphs. *Combinatorics, Probability and Computing*, 3:233–246, 1994. doi:10.1017/S0963548300001139.

11 David P. Dobkin. Computational geometry and computer graphics. *Proc. IEEE*, 80:141–1, 1992.

12 Christian A. Duncan, Emden R. Gansner, Yifan Hu, Michael Kaufmann, and Stephen G. Kobourov. Optimal polygonal representation of planar graphs. *Algorithmica*, 63(3):672–691, 2012. doi:10.1007/s00453-011-9525-2.

13 David Eppstein and Elena Mumford. Steinitz theorems for simple orthogonal polyhedra. *J. Comput. Geom.*, 5(1):179–244, 2014. doi:10.20382/jocg.v5i1a10.

14 William Evans, Paweł Rzążewski, Noushin Saeedi, Chan-Su Shin, and Alexander Wolff. Representing graphs and hypergraphs by touching polygons in 3D. In Daniel Archambault and Csaba Tóth, editors, *Proc. Graph Drawing & Network Vis. (GD'19)*, volume 11904 of *LNCS*, pages 18–32. Springer, 2019. doi:10.1007/978-3-030-35802-0_2.

15 Stefan Felsner. Rectangle and square representations of planar graphs. In János Pach, editor, *Thirty Essays on Geometric Graph Theory*, pages 213–248. Springer, 2013. doi:10.1007/978-1-4614-0110-0_12.

16 Stefan Felsner and Mathew C. Francis. Contact representations of planar graphs with cubes. In Ferran Hurtado and Marc J. van Kreveld, editors, *Proc. 27th Ann. Symp. Comput. Geom. (SoCG'11)*, pages 315–320. ACM, 2011. doi:10.1145/1998196.1998250.

17 Marek Filakovský, Uli Wagner, and Stephan Zhechev. Embeddability of simplicial complexes is undecidable. In *Proc. ACM-SIAM Symp. Discrete Algorithms (SODA)*, pages 767–785, 2020. doi:10.1137/1.9781611975994.47.

18 Emden R. Gansner, Yifan Hu, and Stephen G. Kobourov. On touching triangle graphs. In Ulrik Brandes and Sabine Cornelsen, editors, *Proc. Graph Drawing (GD'10)*, volume 6502 of *LNCS*, pages 250–261. Springer, 2010. doi:10.1007/978-3-642-18469-7.

19 Daniel Gonçalves, Benjamin Lévêque, and Alexandre Pinlou. Homothetic triangle representations of planar graphs. *J. Graph Alg. Appl.*, 23(4):745–753, 2019. doi:10.7155/jgaa.00509.

20 Petr Hliněný. Contact graphs of line segments are NP-complete. *Discrete Math.*, 235(1):95–106, 2001. doi:10.1016/S0012-365X(00)00263-6.

21 Petr Hliněný and Jan Kratochvíl. Representing graphs by disks and balls (a survey of recognition-complexity results). *Discrete Math.*, 229(1–3):101–124, 2001. doi:10.1016/S0012-365X(00)00204-1.

22 Seok-Hee Hong and Hiroshi Nagamochi. Extending steinitz's theorem to upward star-shaped polyhedra and spherical polyhedra. *Algorithmica*, 61(4):1022–1076, 2011. doi:10.1007/s00453-011-9570-x.

23 Lutz Kettner. Designing a data structure for polyhedral surfaces. In *Proc. 14th Annu. ACM Symp. Comput. Geom. (SoCG'98)*, pages 146–154, 1998. doi:10.1145/276884.276901.

24 Tamás Kővari, Vera T. Sós, and Pál Turán. On a problem of K. Zarankiewicz. *Coll. Math.*, 3(1):50–57, 1954. URL: http://eudml.org/doc/210011.

25 Linda Kleist and Benjamin Rahman. Unit contact representations of grid subgraphs with regular polytopes in 2D and 3D. In Christian Duncan and Antonios Symvonis, editors, *Proc. Graph Drawing (GD'14)*, volume 8871 of *LNCS*, pages 137–148. Springer, 2014. doi:10.1007/978-3-662-45803-7_12.

26 Stephen G. Kobourov, Debajyoti Mondal, and Rahnuma Islam Nishat. Touching triangle representations for 3-connected planar graphs. In Walter Didimo and Maurizio Patrignani, editors, *Proc. Graph Drawing (GD'12)*, volume 7704 of *LNCS*, pages 199–210. Springer, 2013. doi:10.1007/978-3-642-36763-2_18.

27 Paul Koebe. Kontaktprobleme der konformen Abbildung. *Berichte über die Verhandlungen der Sächsischen Akad. der Wissen. zu Leipzig. Math.-Phys. Klasse*, 88:141–164, 1936.

28 Jiří Matoušek, Martin Tancer, and Uli Wagner. Hardness of embedding simplicial complexes in $\mathbb{R}^d$. *J. Europ. Math. Soc.*, 13(2):259–295, 2011. doi:10.4171/JEMS/252.

29 P. McMullen, C. Schulz, and J.M. Wills. Polyhedral 2-manifolds in E^3 with unusually large genus. *Israel J. Math.*, 46:127–144, 1983. doi:10.1007/BF02760627.

30 Arnaud de Mesmay, Yo'av Rieck, Eric Sedgwick, and Martin Tancer. Embeddability in $\mathbb{R}^3$ is NP-hard. *J. ACM*, 67(4):1–29, 2020. doi:10.1145/3396593.

31 Jürgen Richter-Gebert. *Realization spaces of polytopes*, volume 1643 of *Lecture Notes in Mathematics*. Springer, 1996. doi:10.1007/BFb0093761.

32 Oded Schramm. *Combinatorically Prescribed Packings and Applications to Conformal and Quasiconformal Maps*. PhD thesis, Princeton University, 2007. arXiv:0709.0710.

33 Arkadiy Skopenkov. Extendability of simplicial maps is undecidable. ArXiv report, 2020. arXiv:2008.00492.

34 Arkadiy Skopenkov. Invariants of graph drawings in the plane. *Arnold Math. J.*, 6:21–55, 2020. doi:10.1007/s40598-019-00128-5.

35 Ernst Steinitz. Polyeder und Raumeinteilungen. In *Encyclopädie der mathematischen Wissenschaften*, volume 3-1-2 (Geometrie), chapter 12, pages 1–139. B. G. Teubner, Leipzig, 1922.

36 Szilassi polyhedron. Wikipedia entry. Accessed 2019-10-08. URL: https://en.wikipedia.org/wiki/Szilassi_polyhedron.

37 Heinrich Tietze. Über das Problem der Nachbargebiete im Raum. *Monatshefte für Mathematik und Physik*, 16(1):211–216, 1905. doi:10.1007/BF01693778.

On Undecided LP, Clustering and Active Learning

Stav Ashur
Department of Computer Science, University of Illinois, Urbana, IL, USA

Sariel Har-Peled
Department of Computer Science, University of Illinois, Urbana, IL, USA

Abstract

We study colored coverage and clustering problems. Here, we are given a colored point set, where the points are covered by k (unknown) clusters, which are monochromatic (i.e., all the points covered by the same cluster have the same color). The access to the colors of the points (or even the points themselves) is provided indirectly via various oracle queries (such as nearest neighbor, or separation queries). We show that one can correctly deduce the color of all the points (i.e., compute a monochromatic clustering of the points) using a polylogarithmic number of queries, if the number of clusters is a constant.

We investigate several variants of this problem, including *Undecided Linear Programming* and covering of points by k monochromatic balls.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Linear Programming, Active learning, Classification

Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.12

Related Version *Full Version*: https://arxiv.org/abs/2103.09308

Funding *Sariel Har-Peled*: Work on this paper was partially supported by a NSF AF award CCF-1907400.

Acknowledgements The authors thank Pankaj Agarwal for useful discussions. We thank Lev Reyzin for pointing out the work by Maass and Turán [15, 16].

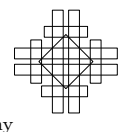
1 Introduction

Given a set of points P in $\mathbb{R}^d$ that are labeled (say, colored as red and blue), the problem of learning a classifier that labels the points correctly is a standard machine learning question. In the active learning settings, querying/exposing the label of an input is an expensive endeavor, and one tries to minimize such queries while performing the learning task.

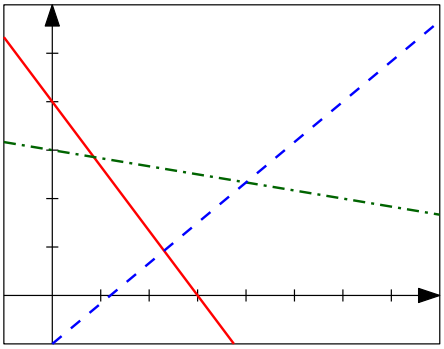
We are interested in a somewhat related question: If the input point set has a "simple" structure, but we are given access to the input via oracles that performs more "interesting" queries than just exposing the label of a point, can one classify correctly all the input points using relatively few oracle queries?

Implicit input model. In particular, consider the situation that instead of the algorithm reading the input, as in the classical settings, the access to the input is via *input primitives*, or *oracles*. Such indirect access to the data rises naturally if the data is already stored in a preexisting database or data-structure. This approach is of relevance nowadays as large amount of data makes even the basic task of reading the input infeasible or prohibitively expensive.

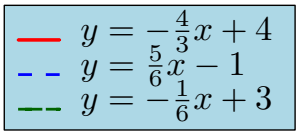
This gives rise to the main motivation for this work – what input primitives/oracles one needs, so that one can derive efficient algorithms. Here, of special interest are algorithms with running times that are sublinear in the input size.



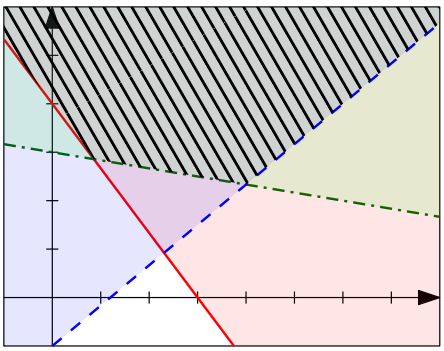

Problem I: Undecided linear programs. An instance of linear programming is a set of n linear inequalities on d variables, where one needs to find an assignment of real values to the variables, such that all the inequalities hold. We consider a new variant of LP, first studied by Maass and Turán [14], where the n linear constraints are given, but we do not know a priori whether the inequality is $\leq$ or $\geq$ for each one of them. Geometrically, this corresponds to being given n hyperplanes in $\mathbb{R}^d$, each having two closed halfspaces associated with it. Which of the two halfspaces is the one used in the LP, can be revealed by querying an oracle. For example, the "standard" *separation oracle*, which returns for a given query point p, a violated constraint of the LP, or alternatively returns that p is feasible.



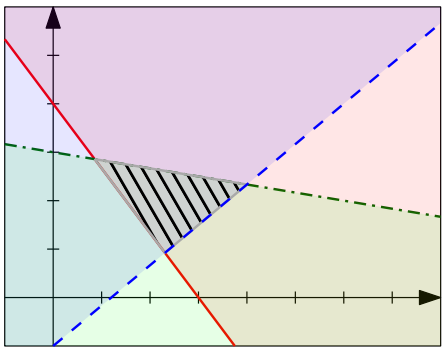
(i) A set of "undecided" constraints in 2d...



(ii) ... is a set of lines in the plane.



(iii) A possible commitment of the constraints, and the feasible polygon induced.



(iv) An alternative commitment of the underlying constraints, and the induced polygon.

Figure 1 An instance of 3 undecided constraints (bottom) with two possible sets of underlying decided constraints (top left and right).

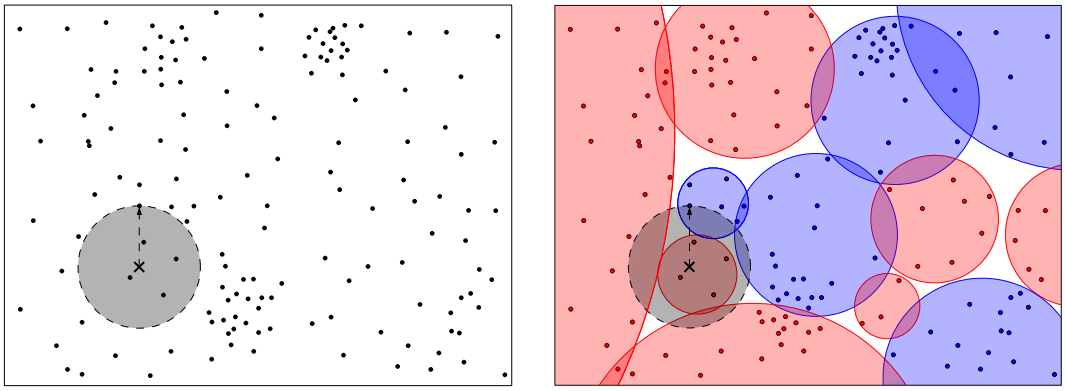
Problem II: Separating red and blue points, with a counterexample oracle. The above problem, in the dual, is the following: The input is a set of unlabeled points, and the task is to compute a hyperplane separating the blue points from the red points. The "separation" oracle here, is given an oriented hyperplane that is supposed to separate the points, and the oracle returns a misclassified point. A natural question is how many queries of this type one has to perform until classifying all the points correctly.

The learning model. Our model seems to be Angluin's equivalence query model for active learning [1]. In particular, Maass and Turán [16, 15] studied the above two problems. See Remark 7 for more details.

Problem III: Covering/clustering points with a ball using proximity oracle. Consider the situation where the input is a set of colored points in $\mathbb{R}^d$, where all the (say) red points are inside a ball, and all the points outside the ball are blue. Here, our access to the color of the points is via an oracle that can answer colored nearest-neighbor (NN) or furthest-neighbor (FN) queries (that is, the oracle can return the closest red [or blue] point to the query point). The task at hand is to label (i.e., color the points) correctly using a minimal number of oracle queries.

The challenge. To appreciate the difficulty in solving the above problem, consider the natural naive algorithm – pick a random sample, expose the colors of the points in the sample (in this case the colored NN queries can do that), compute a ball separating the red points and blue points in the sample, and feed it into the "counterexample" oracle (which returns a colored point that is on the wrong side of the ball – this oracle can be implemented using the NN/FN queries). The algorithm adds this counterexample to the current set of points that their color is known. Now the algorithm repeats the process finding an updated separating circle for the points their colors are known. This algorithm does arrive to the right answer, but it is easy to come up with examples where it has to do a linear number of iterations. As such, the challenge is to get a sublinear number of iterations.

Problem IV: Covering points by monochromatic balls. Consider the situation where the input is a set of points that can be covered by k balls (i.e., clusters). All the points covered by the same ball, have the same label/color (i.e., red or blue). Furthermore, given a query point, the oracle returns the closest point of a prespecified color.



(a) A NN (blue) oracle query marked by an "x" reveals that the points in the interior of the disk are red, and the returned point is blue.

(b) The same query with the underlying colors of the points, and an optimal solution with $\mathsf{k} = 12$ monochromatic disks.

Figure 2 An instance of Problem IV.

Related work. Linear programming has a long history, see the survey by [5].

In *active learning*, also known as query learning or experimental design, the purpose of the algorithm is learning a concept but by querying specific input entries for their label. The main criteria for efficiency is minimizing the number of queries. The basic premise is that asking a specialist to label a specific example is an expensive operation. See Settles [17] for a survey on the topic of active learning.

Table 1 A summary of the results on ULP. Here δ can be chosen to be any constant in $(0, 1)$. The O_d hides constants that depends (probably exponentially or worse) on the dimension.

dim	ref	RT	# queries	Oracle
$d = 2$	Lemma 11	$O(n \log n)$	$O(\log n)$	Separation
	Lemma 5	$O(n)$	$O(\log^2 n)$	
	Lemma 13	$O(n)$	$O(\log n)$	Separation + labeling
$d = 3$	Lemma 15	$\widetilde{O}(n^{3/2})$	$O(\log n)$	Separation
	Theorem 18	$\widetilde{O}(n^{1+\delta})$	$O(\delta^{-1} \log n)$	
d	[16, 15] Lemma 2	$n^{O(d^2)}$	$O(d^2 \log n)$	Separation
	Lemma 2	$\widetilde{O}\left(n^d\right)$	$O(d^3 \log n)$	
	Lemma 5	$O_d(n)$	$O_d(\log^d n)$	
	Remark 12	$O_d(n \log n)$	$O_d(\log^{d-1} n)$	
	Theorem 14	$O_d(n)$	$O_d(\log^{d-1} n)$	Separation + labeling
$d > 3$	Remark 19	$\widetilde{O}\left(n^{1+\delta}\right)$	$O_d(\log^{d-2} n)$	Separation

Closer to our settings, Har-Peled et al. [10], studied algorithms for actively learning a convex body using a separation oracle. Such an oracle either confirms that that the query point is within the convex body, or alternatively, returns a hyperplane separating the point and the convex region.

Kane et al. [13] studied half plane classifiers using comparison queries, and showed an exponential query complexity improvement over learning with only membership queries. Their model is somewhat similar to the model of Problem II above (of separating red and blue points), except that their model assumes that the oracle can return the distance of a query point to the optimal separating hyperplane, while our model is somewhat different, only assuming that the oracle can identify a misclassified point.

In general, computational models that involve various oracles as algorithmic building blocks have been studied in computational geometry, as they represent algorithms in which the input is given implicitly, and access to any information provided by the input is done by oracle queries. See Har-Peled et al. [11] and references therein for such examples.

As mentioned above, Maass and Turán [16, 15] studied Problems I and II – our results are better, but only in constant dimensions, see Remark 7 for details.

1.1 Our results

(A) Undecided LP. We present several algorithms for solving undecided LPs. In Section 2.1, we revisit the algorithm of Maass and Turán, showing ULPs can be solved using $O(d^2 \log n)$ oracle queries, the main tool is repeatedly computing a centerpoint and feeding it to the oracle to further truncate the search space. This bound is polynomial in d, but the algorithm itself is doubly exponential in the dimension d. The running time can be improved to (roughly) $O(n^d)$, with the number of queries deteriorating to $O(d^3 \log n)$.

We present a linear time algorithm, for constant dimension, that uses cutting in Section 2.2, but the number of separation oracle queries is now $O(\log^d n)$.

In Section 2.3, we show that in the plane, if one is allowed also to use an exposure oracle (i.e., an oracle that returns the color of a specific point), then one can reduce the number of oracle queries to $O(\log n)$ (instead of $O(\log^2 n)$ described above). The algorithm is a (potentially interesting) combination of the two previous algorithms.

(B) Covering (red) points by a single ball. In Section 3.1 we study Problem III, we present an algorithm that uses $O(\log^2 n)$ colored NN/FN queries, and computes the single ball that covers (say) all the red points, and avoids all the blue points. The algorithm works by lifting the input point set to three dimensions, and then using the algorithm for undecided LP.

(C) Covering points by k monochromatic balls. In Section 3.2 we address Problem IV above, where the input is covered by k monochromatic balls, and we have access to a colored NN oracle. Inspired by the one ball case, we show a greedy algorithm that finds a ball that covers $O(1/\mathsf{k})$ faction of the uncovered points. This leads to an algorithm that performs $O(k^{d+2}\log^{d+2} n)$ queries (see Theorem 24) and correctly classifies all the points.

2 Algorithms for solving undecided linear programs

▶ Remark 1. All the algorithms described below for undecided LP work in the same fashion – they generate a sequence of separation oracle queries. Specifically, if any of the query points are feasible, the algorithm immediately stops, and output the ULP is feasible, with the queried point as a proof. As such, for the clarity of description, in the following we always assume a separation oracle that returns a separating hyperplane.

2.1 Centerpoint based algorithm for solving ULPs

We review the algorithm of Maass and Turán [16, 15] for solving undecided LP (they did not provide running time bounds for their algorithm, which we do).

Observe that for a set of d-dimensional (closed) hyperplanes H, if there exists a feasible point, then, under general position assumption, there exists a vertex of the arrangement $\mathcal{A}(H)$ that is feasible.

Let P be a set of n points in $\mathbb{R}^d$. For a parameter $\alpha \in (0,1)$, a point $c \in \mathbb{R}^d$ is an **α-centerpoint** if all halfspaces containing c also contain at least αn points of P. A classical implication of Helly's theorem, is that for any set P of n points in $\mathbb{R}^d$, there is a $1/(d+1)$-centerpoint. Such a point is simply a **centerpoint** of P. Such a centerpoint can be computed in $O(n^{d-1})$ time [12, 3]. A $2/d^2$-centerpoint can be computed in near linear time [8] (here, the running time is polynomial in d).

The algorithm. The input is a set H of n hyperplanes in d dimensions. The first step of the algorithm is to compute the set P of vertices of the arrangement $\mathcal{A}(H)$. The number of such vertices is $\leq \binom{n}{d}$. As long as P has more than $d^2 \log n$ points, the algorithm computes a centerpoint c of P. The algorithm then queries the separation oracle on c to decide whether c is feasible. If it is, then the algorithm is done, as it computed a feasible point. Otherwise, the oracle returned a violated constraint of the given LP (this also provides the algorithm with the direction of the constraint for the associated input hyperplane). The algorithm removes all the points of P that violate this constraint, and repeats until $|P| = O(d^2 \log n)$. Once P is that small, the algorithm simply checks the feasibility of each of the remaining vertices by querying it with the separation oracle.

▶ **Lemma 2** (proof in full version [2]). *The above algorithm computes a feasible point of H using $O(d^2 \log n)$ separation oracle queries. The runtime of this algorithm, ignoring the oracle calls, is $O(n^{d(d-1)})$ time.*

Alternatively, the algorithm can be modified so that it computes a feasible point using $O(d^3 \log n)$ separation oracle queries. The running time of this algorithm, ignoring the oracle calls, is $\widetilde{O}(n^d)$ time, where $\widetilde{O}$ hides polylogarithmic terms.

2.2 Cutting based algorithm

Here, we present the new algorithm to solved undecided LP using cuttings.

▶ **Definition 3.** *For a set of n d-dimensional hyperplanes H, a **$1/r$-cutting** of H is a partition Ξ of $\mathbb{R}^d$ into $O(r^d)$ simplices, such that the interior of each simplex of the cutting intersects at most n/r hyperplanes of H. The list of hyperplanes intersecting the interior of a simplex $\nabla \in \Xi$, is the **conflict list** of ∇.*

A $1/r$-cutting, and its conflict list can be computed in $O(nr^{d-1})$ time [7].

▶ Remark 4. The algorithm we present next call recursively on subsets of the constraints, and it also calls recursively on lower dimensional subspaces. In particular, the oracle can be applied to any lower dimensional affine subspace F, by using the original oracle in the ambient space – a returned constraint can be intersected with F to get a constraint in F. We emphasize that the oracle always work on the whole original input set of constraints – the recursive calls on subsets of the constraints are done for efficient bookkeeping, and do not effect how the oracle works.

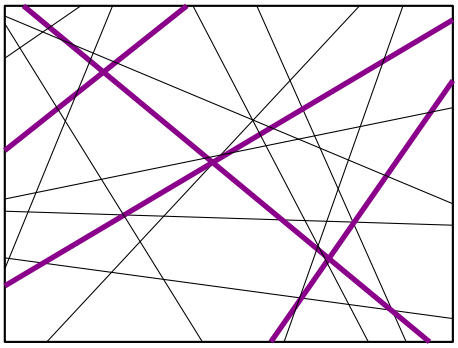
The Algorithm. The algorithm computes a $1/r$-cutting of H. This is a partition of $\mathbb{R}^d$ into $O(r^d)$ simplices, such that each simplex intersects at most n/r of the hyperplanes of H, where $n = |H|$. Let Ξ be this set of simplices, and let $\Xi_{d-1}(\Xi)$ be the set of $O(r^d)$ $(d-1)$-dimensional simplices that form the faces of the simplices of Ξ. The algorithm now solves the problem recursively on each of the $(d-1)$-dimensional simplices of $\Xi_{d-1}(\Xi)$, and the hyperplanes of H that intersects it. Each recursive call is on a problem that is one dimensional lower, and involves only n/r constraints, the oracle still applies to the whole set of constraints), see Remark 4.

If any of these recursive calls finds a feasible point, then we are done. Otherwise, the recursive calls performed involved the oracle, and forced some of the constraints to expose themselves. Let $\mathcal{C}$ be the set of these committed halfspaces (i.e., all the halfspaces returned by the separation oracle). If the intersection of all these constraints is empty (i.e., the associated LP is infeasible), then this can be discovered, in $O(|\mathcal{C}|)$ time, by invoking a standard LP solver on $\mathcal{C}$. If the LP is feasible, then it returns us a point p inside the polytope

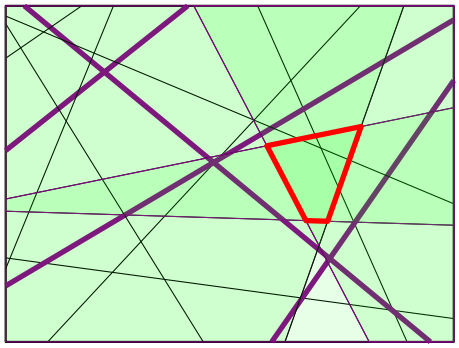
$$\mathcal{K} = \bigcap_{h^+ \in \mathcal{C}} h^+.$$

Furthermore, this polytope must be fully contained in the interior of one of the simplices of Ξ (otherwise, the algorithm would have found a feasible point in one of the recursive calls). By scanning Ξ, we discover the simplex $\nabla \in \Xi$ that contains p. The algorithm now call recursively on ∇ and its conflict list (passing $\mathcal{C}$ as the current set of committed constraints).

Algorithm in one dimension. The 1-dimensional case is solved using a binary search. Indeed, we have n uncommitted rays on the real line, and our purpose is to find an atomic interval that is feasible. To this end, the algorithm computes a median among the points



Cells of a cutting of an arrangement of undecided constraints.



A single cell of the cutting might contain a possibly feasible region in its interior.

Figure 3 Illustration of an iteration of the cutting-based algorithm. A cutting of an arrangement of undecided constraints is depicted on the left. After solving the problem recursively on the cuttings of the edges, some of the constraints are now committed. Their feasible region (the red polygon) they induce is now a polygon that must be fully contained in one of the cells of the cutting.

defining the rays. The algorithm then asks the oracle to commit the direction of the ray. This decrease the number of potential atomic intervals that might be feasible by half. In the end of the process, the algorithm is left with a single atomic interval that might be feasible – the algorithm asks the oracle whether the middle of this interval is feasible or not.

As such, after $O(\log n)$ iterations, the algorithm is done, as each iteration either finds a feasible point, or throws away half the rays as being irrelevant. The running time is $O(n)$, and the number of oracle queries performed is $O(\log n)$.

Analysis: Number of oracle queries. The query complexity of this algorithm is

$$Q_d(n) = O(r^d) Q_{d-1}\Big(\frac{n}{r}\Big) + Q_d\Big(\frac{n}{r}\Big),$$

with $Q_1(n) = O(\log n)$. The solution of this recurrence is $O(\log^d n)$, for r chosen to be a sufficiently large constant. Indeed, using induction, we have $Q_d(n) = O(r^d \log^{d-1} \frac{n}{r}) + Q_d\big(\frac{n}{r}\big)$,

Running time. As for the running time, we have

$$T_d(n) = O(nr^{d-1}) + O(r^d) T_{d-1}(n/r) + T_d(n/r),$$

with $T_1(n) = O(n)$. Assuming $T_{d-1}(n) \leq c_{d-1} n$, and for two constants c'_d and c''_d, we have

$$T_d(n) \leq c'_d n r^{d-1} + c''_d r^d c_{d-1} \frac{n}{r} + c_d \frac{n}{r} \leq \Big(c'_d r^{d-1} + c''_d r^{d-1} c_{d-1} + \frac{c_d}{r}\Big) n \leq c_d n,$$

which holds if $c_d \geq 2(c'_d r^{d-1} + c''_d r^{d-1} c_{d-1})$. This implies that $T_d(n) = O(n)$. We thus get the following result.

▶ **Lemma 5.** *Undecided LP with n constraints in $\mathbb{R}^d$, can be solved in $O_d(n)$ time, using $O(\log^d n)$ separation oracle queries.*

▶ Remark 6 (An implicit undecided LP). In the following, we need a variant of the above problem – the input is a set of undecided constraints, but the real undecided LP instance $\mathcal{I}$ corresponds to an (unknown) subset of these constraints. Fortunately, the given separation oracle works on the "real" instance of constraints $\mathcal{I}$. It is easy to verify that the above algorithm of Lemma 5 works verbatim in this case.

▶ Remark 7. The work of Maass and Turán [16, 15] also studied the dual settings of our problem (i.e., Problem II of separating red and blue points) They assume the input points are taken from a grid of bounded spread. They use the ellipsoid algorithm to get an efficient algorithm with small number of queries.

2.3 Better algorithms in two dimensions

The algorithm of Lemma 5 above uses only a separation oracle. One can get a faster algorithm if one is allowed to use a **labeling oracle** – this oracle, given an input point, returns its label/color. In our case, the labeling oracle reveals the underlying halfspace constraint defined by (undecided) input hyperplane.

We need the following lemma, due to Har-Peled and Mitchell [9].

▶ **Lemma 8** ([9]). *Let D be a fixed polygon with k edges, and let L be a set of m lines that intersect the interior of D. After $O(m(\log k + \log m))$ preprocessing, one can compute the number of vertices of $\mathcal{A}(L)$ that lie inside D, or sample such a vertex in $O(\log m)$ time.*

2.3.1 Polygon and point set reduction

▶ **Lemma 9** (proof in full version [2]). *Consider an instance of undecided LP in the plane, and a polygon D with t edges, such that the feasible region is contained in D. Using $O(\log t)$ separation queries, and $O(t)$ time, one can either compute a feasible point, or compute a polygon $D' \subseteq D$ with (say) at most 10 edges, such that the feasible region must be contained in D'.*

▶ **Lemma 10** (proof in full version [2]). *Consider an instance of undecided LP in $\mathbb{R}^d$, and a set P of n points. In $O(n)$ time, using $O(\log n)$ separation oracle queries, one can compute either: (i) a feasible point to the ULP, or alternatively, (ii) a polytope D with $O(\log n)$ faces, such that the feasible solution for the ULP lies inside D, and D contains no point of P.*

2.3.2 Near linear running time in two dimensions

Let L be the input set of n lines (i.e., undecided constraints). Let $L_0 = L$, and $D_0 = \mathbb{R}^2$.

Let V_i denote the set of vertices of $\mathcal{A}(L_{i-1})$ that lie in D_{i-1}. The algorithm computes $n_i = |V_i|$, using the algorithm of Lemma 8. There are two possibilities:

(A) If $n_i = 0$, then the algorithm picks any point p_i in D_{i-1}, and calls the separation oracle on p_i. If p_i is feasible the algorithm is done, otherwise, the algorithm returns that the given instance is infeasible.

(B) If $n_i = O(n \log n)$, the algorithm computes V_i by clipping the lines of L to D_{i-1}, and computing the arrangement of the resulting segments. This takes $O(n \log n + n_i)$ expected time, as this the time for computing the arrangement of n segments with n_i intersections.

The algorithm uses Lemma 10 on V_i to either find a feasible point, or a polygon D'_1 that must contain the feasible region, has at most $O(\log n)$ edges, and contains no point of V_i. The algorithm next uses Lemma 9 to further reduce the polygon into a polygon $D_i \subseteq D'_i$ with constant number of edges.

(C) Otherwise, the algorithm samples $O(n)$ vertices from V_i, and let R_i be the resulting sample. This is done using the algorithm of Lemma 8 in $O(n \log n)$ time. Next, the algorithm applies, as above, Lemma 10 and Lemma 9 to get a constant complexity polygon $D_i \subseteq D_{i-1}$ that does not contain any of the points of R_i.

The algorithm now scans the lines of L_{i-1}, computes the set of all lines $L_i \subseteq L_{i-1}$ that intersect D_i, and continues to the next iteration.

▶ **Lemma 11** (proof in full version [2]). *The above algorithm solves two dimensional undecided LP in expected $O(n \log n)$ time, using $O(\log n)$ separation oracle queries.*

▶ Remark 12. Combining the algorithm of Lemma 5 together with the algorithm of Lemma 11, when the dimension is two, results in an algorithm that solves ULP in d dimensions, with $O(\log^{d-1} n)$ separation queries, and running time $O(n \log n)$.

2.3.3 A linear time algorithm (using labeling oracle)

In the ith iteration, the algorithm computes a polygon D_i, with at most $k = 10$ boundary edges, that might contain a feasible point, and a list of lines L_i that intersect it in the ith iteration. Initially, D_0 is the whole plane, and $L_i = H$.

In the beginning of the ith iteration, the algorithm computes a random sample $R_i \subseteq L_i$ of size $O(\varepsilon^{-1} k \log k \log \varepsilon^{-1}) = O(1)$, where $\varepsilon = 1/2$. The sample R_i is, with probability close to one, an ε-net of L_i for polygons that have at most k boundary edges. The algorithm now calls the oracle for each undecided constraint of R_i to expose its true constraint. Let F_i be the face of the arrangement of the revealed constraints that is still feasible (if no such face exists, then we are done). The algorithm next considers the polygon $D'_i = F_i \cap D_{i-1}$. If this polygon is empty, then the algorithm is done. If D'_i has at most k edges, then the algorithm sets $D_i = D'_i$, computes $L_i = L_{i-1} \cap D_i$, and continues to the next iteration.

The only bad case here, is that D'_i has too many edges. Clearly, in the worst case, it can have at most $|R_i| + k$ edges. The algorithm now computes a centerpoint for the vertices of this polygon, and sends it to the oracle. If the point is feasible, then the algorithm is done. Otherwise, the oracle returned a violated constraint, and the polygon is split into two polygons, each with at most two thirds of its original vertices. Namely, in the new feasible polygon the number of edges is decreased by a constant factor. After a constant number of such iterations, the feasible polygon has at most k edges. The algorithm then sets this polygon to be D_i, and continues to the next iteration, as described above.

if L_i is empty, then the algorithm asks the oracle if some point in the interior of the polygon is feasible, and if not the given ULP is not feasible.

Analysis. Observe that D_i intersects no lines of R_i, and it has at most k edges. As such, by the ε-net theorem, with constant probability (say ≥ 0.99), the interior of D_i intersects at most $\varepsilon |L_{i-1}|$ lines. Namely, $|L_i| \leq |L_{i-1}|/2$. This implies that the algorithm performs, in expectation (and also with high probability), at most $O(\log n)$ iterations. Each iteration requires $O(1) + O(|R_i|) = O(1)$ oracle queries, which readily implies a bound of $O(\log n)$ oracle queries.

The correctness of the algorithm itself follows as the feasible region if the LP is always contained inside D_i.

As for running time, each iteration requires $O(1 + |L_i|)$ amount of work. As such, the total expected running time is $\sum_i O(1 + |L_i|) = O(n)$.

▶ **Lemma 13** (proof in full version [2]). *Undecided LP with n constraints in $\mathbb{R}^2$, can be solved in $O(n)$ expected time, using $O(\log n)$ separation and labeling oracle queries.*

Combining the above algorithm with the cutting based algorithm of Lemma 5, results in the following slightly improved algorithm.

▶ **Theorem 14.** *Undecided LP with n constraints in $\mathbb{R}^d$, for $d \geq 2$, can be solved in $O_d(n)$ time, using $O(\log^{d-1} n)$ separation and labeling oracle queries.*

2.4 A query efficient algorithm in three dimensions

2.4.1 Emulating the two dimensional algorithm

In three dimensions, one can still get $O(\log n)$ queries, albeit getting running time $\tilde{O}(n^{3/2})$.

The input is a set H of n planes in three dimensions. The algorithm randomly samples a set V_1 of $O(n^{3/2}\log^3 n)$ vertices of $\mathcal{A}(H)$. Each vertex is generated by randomly choosing three constraints (planes) from H, and computing their intersection. Using the algorithm of Lemma 10, one computes a convex polytope $\mathcal{K}_1$ with $O(\log n)$ faces, which does not contain any vertex of V_1 (since Lemma 10 returns the $O(\log n)$ halfspaces whose intersection form the desired polytope, the polytope itself can be computed in $O(\log n \log\log n)$ time).

Next, the algorithm computes all the vertices of $\mathcal{A}(H)$ inside $\mathcal{K}_1$ – this can be done in an output sensitive fashion. To this end, one "walks" around the arrangement of $\mathcal{A}(H)$, starting (say) with the bottom vertex of $\mathcal{K}_1$. Specifically, one walks on edges of the arrangement (inside $\mathcal{K}_1$) using the data-structure of Chan [4], which provide dynamic maintenance of convex-hull in three dimensions, and extreme point queries. Each operation takes $O(\log^4 n)$ amortized time. The exact details of this exploration are somewhat delicate, but straightforward, and we omit them as they are similar in nature to the 2d algorithms (see [6] and references therein). Let V_2 be the resulting set of vertices. As we show below, with high probability $|V_2| = O(n^{3/2})$. As such, computing this set takes $O(n^{3/2}\log^4 n)$ time.

Deploying the algorithm of Lemma 10, one computes a convex polytope $\mathcal{K}_2$ with $O(\log n)$ faces, which does not contain any vertex of V_2. Consider the intersection polytope $\mathcal{K} = \mathcal{K}_1 \cap \mathcal{K}_2$. It avoids the vertices of V_1 and V_2 and thus avoids all the vertices of the arrangement of $\mathcal{A}(H)$. Furthermore, $\mathcal{K}$ is formed by the intersection of halfspaces, all induces by planes of H. It follows that $\mathcal{K}$ is a 3d face of the arrangement. As such, perform a single query on a point in the interior of $\mathcal{K}$, and return it as the output of the algorithm.

The algorithm again invokes Lemma 10 to compute a polytope $\mathcal{K}_2$ that contains no vertex of V_2. The algorithm computes the polytope $\mathcal{K}_1 \cap \mathcal{K}_2$ (in polylogarithmic time). If the intersection is empty then the given instance is infeasible. Otherwise, the algorithm query an point inside this intersection using the separation oracle. If the point is feasible then the algorithm is done, and otherwise, again, the instance is infeasible.

▶ **Lemma 15** (proof in full version [2]). *The above algorithm solves Undecided LP with n constraints, in $\mathbb{R}^3$, in $O\left(n^{3/2}\log^3 n\sqrt{\log\log n}\right)$ time, using $O(\log n)$ separation oracle queries.*

2.4.2 A faster algorithm

In the following, we assume that the set of planes of H is in general position – no three planes passes through a common line, and no four planes have a common intersection point.

Idea. A natural approach for getting a faster algorithm is to maintain a polytope $\mathcal{K}_i$, sample an ε-net for the vertices of the arrangement inside $\mathcal{K}_i$ (for an ε to yet be specified), and use the algorithm of Lemma 9 to find a low complexity polytope that avoids all the points in this ε-net. Intersecting this polytope with the previous active polytope, results in a shrunken feasible region $\mathcal{K}_i$. Furthermore, $\mathcal{K}_i$ contains an ε-fraction of the vertices of the arrangement inside it compared to $\mathcal{K}_{i-1}$. The algorithm then continues to the next iteration, till the polytope contains no vertices of $\mathcal{K}_i$, and then a single query in its interior settles the feasibility of the given ULP.

The challenge is that despite $\mathcal{K}_i$ being simple (i.e., having few faces), we do not know how to sample uniformly and efficiently from $\mathcal{V} \cap \mathcal{K}_i$, where $\mathcal{V} = \mathcal{V}(H)$ is the set of vertices of $\mathcal{A}(H)$. We leave this an open problem for further research. Instead, we offer the following over-sampling approach.

▶ **Lemma 16** (proof in full version [2]). *Let $\mathcal{K}$ be a polytope in three dimensions with k faces, and let H be a set of n planes, where all the faces of $\mathcal{K}$ lie on planes of H. One can sample a non-empty set X of at most $n-1$ vertices, such that (i) $X \subseteq \mathcal{V} \cap \mathcal{K}$, where $\mathcal{V} = \mathcal{V}(H)$, and the probability of any vertex of $\mathcal{V} \cap \mathcal{K}$ to be included in the sample is the same.*

The preprocessing time of the algorithm is $O(nk \log n)$, and a sampled set can be computed in $O(n)$ time.

▶ **Lemma 17** (proof in full version [2]). *Let $\delta \in (0,1)$ be a fixed constant, let $\mathcal{K}$ be a polytope in three dimensions with $t = O(\log n)$ faces, and let H be a set of n planes, where all the faces of $\mathcal{K}$ lie on planes of H. Let $\mathcal{V} = \mathcal{V}(H) \cap \mathcal{K}$ be the set of vertices of $\mathcal{A}(H)$ that lie inside $\mathcal{K}$. One can compute a polytope $\mathcal{K}'$, that is the intersection of $\mathcal{K}$ with $O(\log n)$ halfspaces, such that $\mathcal{K}'$ contains at most $|\mathcal{V}|/n^{\delta}$ vertices of $\mathcal{A}(H)$. The algorithm runs in $O(n^{1+\delta} \log^3 n \log\log n)$ time. The algorithm uses $O(\log n)$ separation oracle queries.*

Starting with $\mathbb{R}^3$ as the initial polytope, the algorithm repeatedly uses Lemma 17 to reduce the number of vertices of the arrangement inside the current polytope by a factor of $1/n^{\delta}$. In the ith iteration, the current polytope has $O(i \log n)$ faces, and as such the final polytope has at most $O((3/\delta)\log n)$ faces, as the algorithm has no vertices in it after $\lceil 3/\delta \rceil$ iterations. We conclude the following.

▶ **Theorem 18.** *For any $\delta \in (0,1)$, an instance of undecided LP in three dimensions with n constraints, one can solved using $O((\log n)/\delta)$ separation oracle queries, in*

$$O(n^{1+\delta} \log^4 n \log\log n)$$

time.

▶ Remark 19. Combining the algorithm of Lemma 5, together with the algorithm of Theorem 18, when the dimension is three, results in an algorithm that solves ULP in $d > 3$ dimensions, with $O(\delta^{-1} \log^{d-2} n)$ separation queries, and running time $\widetilde{O}(n^{1+\delta})$.

3 Covering points by monochromatic balls using proximity queries

3.1 Learning a single monochromatic ball using NN/FN queries

Problem statement. The input is a set $P = \{p_1, ..., p_n\}$ of n points in $\mathbb{R}^d$. The points are either blue or red, but their color is not initially provided. We have an access to a nearest-neighbor (NN) oracle, such that given a query point, and a color, it returns the closest point of this color to the query point. Similarly, we are given access to a furthest-neighbor (FN) oracle, that returns the furthest point of this color in P.

The task at hand is to correctly classify all the given points as either red or blue, minimizing the number of queries used.

Assumption: Single ball. Here we assume that all the red points in P can be covered by a single ball, while all the blue points are outside this ball. Our purpose here is to develop an efficient algorithm that performs as few oracle queries as possible, and exposes the color of all the points of P.

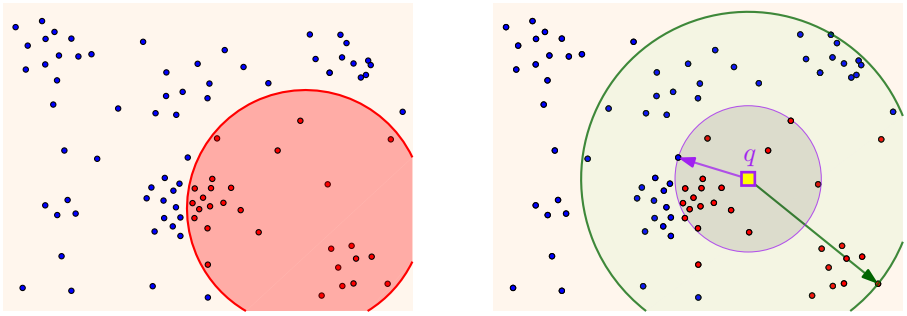


Figure 4 A set of red and blue points in $\mathbb{R}^2$ where a single disk suffices to separate the labels. A red furthest-neighbor (FN) query and a blue nearest-neighbor (NN) query at q confirms that a monochromatic disk centered at that point cannot contain every red point of the input.

The algorithm is described in the plane, but it also works in higher dimensions with minor modifications.

3.1.1 Lifting to three dimensions

Let $\text{disk}(p, r)$ denote the disk of radius r centered at p. Consider the mapping of such a disk to the point

$$\mathcal{d} = \vartheta(\text{disk}(p, r)) = \left(2p_x, 2p_y, r^2 - p_x^2 - p_y^2\right).$$

This can be interpreted as a somewhat bizarre encoding of disks as points in three dimensions. We also map a point $q \in \mathbb{R}^2$ in the plane, to the plane

$$\varphi(q) = \left(z = -q_x x - q_y y + q_x^2 + q_y^2\right).$$

Note that if $\mathcal{d}$ is above $h = \varphi(q)$ then

$$\mathcal{d}_z \geq -q_x \mathcal{d}_x - q_y \mathcal{d}_y + q_x^2 + q_y^2. \iff r^2 - p_x^2 - p_y^2 \geq -2q_x p_x - 2q_y p_y + q_x^2 + q_y^2.$$
$$\iff r^2 \geq \left(p_x - q_x\right)^2 + \left(p_y - q_y\right)^2 \iff q \in \text{disk}(p, r).$$

3.1.2 The algorithm

The above lifting of disks to points (in three dimensions), and points to planes, has the property that a point is above a plane $\iff$ the original disk contains the original point. In particular, if a lifted disk $\mathcal{d} \in \mathbb{R}^3$ is strictly below a plane h, then the original disk does not contain q (i.e., the original point lifted to h).

In the lifted space, the input is a set of n planes in three dimensions. If a plane h is red, then the computed disk must contain the original point, which means that the encoded disk $\mathcal{d}$ must lie above h. Namely, every red point, corresponds to a commitment of the corresponding plane, to the halfspace lying (vertically) above it. Similarly, an original blue point corresponds to a commitment to the downward halfspace. We want to find a point in this space which is feasible (after all the constraints have been committed).

The ULP oracle queries. A labeling oracle query on a plane, corresponds to providing the color of the original point, which can be done using a single NN colored query. A feasibility oracle query, is a point $\mathcal{d}$ in three dimensions, which corresponds to a disk, which asks

whether it contains all the red points, and no blue points. The later can be answered by performing a blue NN query, and a red FN query, and then making a decision according to how the points interact with the query disk.

As such, we can plug the lifted instance into the algorithm of Theorem 14. This algorithm also works verbatim in higher dimensions. We thus get the following.

▶ **Theorem 20.** *Let P be a set of n points in $\mathbb{R}^d$. Assume that there is an underlying coloring of the point set by (say) red and blue, and there is an oracle that can answer nearest-neighbor and furthest-neighbor colored queries on P. The above algorithm computes a ball that contains only the red points, and no blue points, if such a ball exists, using $O(\log^{d-1} n)$ NN/FN oracle queries. The running time of the algorithm is $O_d(n)$.*

3.2 Learning a cover by k monochromatic balls using NN queries

Problem statement. The input is a set of n colored points P, and assume that the points of P can be covered by k balls, such that each ball covers only points of a single color. Here, we assume that we are given access to the points via a NN oracle queries (i.e., no FN queries). The task at hand is to classify the points correctly (i.e., decide their color) using a small number of oracle queries, by an efficient algorithm.

3.2.1 The algorithm

▶ **Lemma 21** (proof in full version [2]). *Let Q be a set of m points in $\mathbb{R}^d$, and let $\mathcal{R}$ be the (infinite) set of all balls in $\mathbb{R}^d$. One can compute, in $O(m^{d+2})$ time, the family of $O(m^{d+1})$ canonical sets induced by $\mathcal{R}$ on Q, that is $\mathcal{F}(Q) = \{\mathbb{b} \cap Q \mid \mathbb{b} \in \mathcal{R}\}$.*

For $i > 0$, let P_i denote the set of unlabeled points at the beginning of the ith iteration and let $n_i = |P_i|$ (i.e., $P_1 = P$, and $n_1 = n$). The algorithm computes a random sample $R_i \subseteq P_i$ of size $O(\mathsf{k} \log \mathsf{k})$, and determines the color of the points in R_i using NN queries. The algorithm then computes the set of canonical sets $\mathcal{F}_i = \mathcal{F}(R_i)$, using Lemma 21. For every range $\mathbf{r} \in \mathcal{F}_i$, such that $\overline{s}_{R_i}(\mathbf{r}) = \dfrac{|\mathbf{r} \cap R_i|}{|R_i|} \geq \dfrac{1}{2\mathsf{k}}$, and such that all the points in $\mathbf{r}$ are of the same color, the algorithm runs a subroutine, described below in Section 3.2.2, to decide if there is a monochromatic ball that contains all the points of $\mathbf{r}$. Formally, the subroutine decides if there is ball $\mathbb{b}$, such that all the points of $P \cap \mathbb{b}_i$ are colored by the same color, and $\mathbf{r} \subseteq \mathbb{b}$. If no such ball is found, the algorithm repeats this iteration until success.

The algorithm sets $\mathbb{b}_i$ to be the ball computed, such that $|\mathbb{b}_i \cap P_i|$ is maximized among all such balls. The algorithm then adds $\mathbb{b}_i$ to the computed cover, assigns all the points in $\mathbb{b}_i \cap P_i$ their color, and sets $P_{i+1} \leftarrow P_i \setminus \mathbb{b}_i$.

The algorithm stops once all points have been assigned their correct color (i.e., $P_i = \emptyset$).

3.2.2 Searching for a monochromatic ball containing a set r

The subroutine is given a set of points $\mathbf{r}$ that are all (say) red. The subroutine is searching for a ball $\mathbb{b}$ such that the point in $P \cap \mathbb{b}$ are all red points, and $\mathbf{r} \subseteq \mathbb{b}$. The subroutine is similar in spirit to the single ball case of Section 3.1, but the details are somewhat different.

Specifically, consider the set B of all blue points in P (this set is not explicitly known, as many of the points color is yet unknown), and consider the problem of computing a ball that contains all the points of $\mathbf{r}$ and none of the points of B. This is an implicit undecided optimization problem, which via the lifting of Section 3.1.1, reduces to implicit undecided LP.

A separation oracle here, in the original settings, is a query ball $\mathcal{q}$. If $\mathcal{q}$ contains a blue point, one can find it by performing a colored nearest-neighbor query on the blue points. Similarly, one can verify that $\mathcal{b}$ contains all the red points of $\mathbf{r}$. Thus, one can use the implicit undecided LP algorithm of Lemma 5 (see Remark 6).

3.2.3 Analysis

Informally, each successful iteration reveals the color of a $\Omega(1/\mathsf{k})$-fraction of the unlabeled points. Specifically, at the ith iteration, at least one of the k balls of the optimal solution must contain at least n_i/k points of P_i, which are all of the same color (and are yet unlabeled). As such, after $O(\mathsf{k} \log n)$ iterations, the algorithm correctly exposes the colors of the points in P.

▶ **Lemma 22** (proof in full version [2]). *Given a set $\mathbf{r} \subseteq P$ of (say) red points, such that there exists a ball $\mathcal{b}$ such that all the points of $\mathcal{b} \cap P$ are of the same color, and $\mathbf{r} \subseteq \mathcal{b}$, the subroutine of Section 3.2.2 returns a monochromatic ball that covers the points of $\mathbf{r}$.*

▶ **Lemma 23** (proof in full version [2]). *Under the assumption that the input can be covered by k monochromatic balls, an iteration of the above algorithm succeeds with probability close to one, and computes a monochromatic ball that covers at least $2/(5\mathsf{k})$ fraction of the uncolored points.*

▶ **Theorem 24** (proof in full version [2]). *Let P be a set of n points $\mathbb{R}^d$, such that there are (unknown) k monochromatic balls that cover all the points of P. Furthermore, assume we are given an oracle that can answer NN colored queries on P. Then, one can compute the color of all the points of P using $O\Big(\mathsf{k}^{d+2} \log^{2d+3} n\Big)$ queries (this bound holds in expectation). The expected running time of the algorithm is $O\Big(\mathsf{k}^{d+2} n \log^{d+1} \mathsf{k}\Big)$.*

References

1. Dana Angluin. Queries and concept learning. *Mach. Learn.*, 2(4):319–342, 1987. doi:10.1007/BF00116828.
2. Stav Ashur and Sariel Har-Peled. On undecided LP, clustering and active learning. *CoRR*, abs/2103.09308, 2021. arXiv:2103.09308.
3. Timothy M. Chan. An optimal randomized algorithm for maximum Tukey depth. In J. Ian Munro, editor, *Proc. 15th ACM-SIAM Sympos. Discrete Algs.* (SODA), pages 430–436, Philadelphia, PA, USA, 2004. SIAM. URL: http://dl.acm.org/citation.cfm?id=982792.982853.
4. Timothy M. Chan. Dynamic geometric data structures via shallow cuttings. In *Proc. 35th Int. Annu. Sympos. Comput. Geom.* (SoCG), pages 24:1–24:13, 2019. doi:10.4230/LIPIcs.SoCG.2019.24.
5. Martin E. Dyer, Nimrod Megiddo, and Emo Welzl. Linear programming. In Jacob E. Goodman and Joseph O'Rourke, editors, *Handbook of Discrete and Computational Geometry, Second Edition*, pages 999–1014. Chapman and Hall/CRC, 2004. doi:10.1201/9781420035315.pt6.
6. S. Har-Peled. Taking a walk in a planar arrangement. *SIAM J. Comput.*, 30(4):1341–1367, 2000.
7. S. Har-Peled. *Geometric Approximation Algorithms*, volume 173 of *Math. Surveys & Monographs*. Amer. Math. Soc., Boston, MA, USA, 2011. doi:10.1090/surv/173.
8. Sariel Har-Peled and Mitchell Jones. Journey to the center of the point set. In Gill Barequet and Yusu Wang, editors, *Proc. 35th Int. Annu. Sympos. Comput. Geom.* (SoCG), volume 129 of *LIPIcs*, pages 41:1–41:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2019. doi:10.4230/LIPIcs.SoCG.2019.41.

9 Sariel Har-Peled and Mitchell Jones. On separating points by lines. *Discret. Comput. Geom.*, 63(3):705–730, 2020. doi:10.1007/s00454-019-00103-z.

10 Sariel Har-Peled, Mitchell Jones, and Saladi Rahul. Active learning a convex body in low dimensions. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *Proc. 47th Int. Colloq. Automata Lang. Prog.* (ICALP), volume 168 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 64:1–64:17, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2020.64.

11 Sariel Har-Peled, Nirman Kumar, David M. Mount, and Benjamin Raichel. Space exploration via proximity search. *Discrete Comput. Geom.*, 56(2):357–376, 2016. doi:10.1007/s00454-016-9801-7.

12 S. Jadhav and A. Mukhopadhyay. Computing a centerpoint of a finite planar set of points in linear time. *Discrete Comput. Geom.*, 12:291–312, 1994.

13 Daniel M. Kane, Shachar Lovett, Shay Moran, and Jiapeng Zhang. Active classification with comparison queries. In *Proc. 58th Annu. IEEE Sympos. Found. Comput. Sci.* (FOCS), pages 355–366, 2017. doi:10.1109/FOCS.2017.40.

14 Wolfgang Maass and György Turán. On the complexity of learning from counterexamples and membership queries. In *Proc. 31st Annu. IEEE Sympos. Found. Comput. Sci.* (FOCS), pages 203–210. IEEE Computer Society, 1990. doi:10.1109/FSCS.1990.89539.

15 Wolfgang Maass and György Turán. Algorithms and lower bounds for on-line learning of geometrical concepts. *Machine Learning*, 14(3):251–269, 1994. doi:10.1007/BF00993976.

16 Wolfgang Maass and György Turán. How fast can a threshold gate learn? In *Proc. Work. Comput. Learning Theory and Nat. Learn. Systems*, pages 381—414, Cambridge, MA, USA, 1994. MIT Press.

17 Burr Settles. Active learning literature survey. Technical Report #1648, Computer Science, Univ. Wisconsin, Madison, 2009. URL: https://minds.wisconsin.edu/bitstream/handle/1793/60660/TR1648.pdf?sequence=1&isAllowed=y.

Two-Sided Kirszbraun Theorem

Arturs Backurs
Toyota Technological Institute at Chicago (TTIC), IL, USA

Sepideh Mahabadi
Toyota Technological Institute at Chicago (TTIC), IL, USA

Konstantin Makarychev
Northwestern University, Evanston, IL, USA

Yury Makarychev
Toyota Technological Institute at Chicago (TTIC), IL, USA

Abstract

In this paper, we prove a two-sided variant of the Kirszbraun theorem. Consider an arbitrary subset X of Euclidean space and its superset Y. Let f be a 1-Lipschitz map from X to $\mathbb{R}^m$. The Kirszbraun theorem states that the map f can be extended to a 1-Lipschitz map $\tilde{f}$ from Y to $\mathbb{R}^m$. While the extension $\tilde{f}$ does not increase distances between points, there is no guarantee that it does not decrease distances significantly. In fact, $\tilde{f}$ may even map distinct points to the same point (that is, it can infinitely decrease some distances). However, we prove that there exists a $(1+\varepsilon)$-Lipschitz outer extension $\tilde{f} : Y \to \mathbb{R}^{m'}$ that does not decrease distances more than "necessary". Namely,

$$\|\tilde{f}(x) - \tilde{f}(y)\| \geq c\sqrt{\varepsilon} \min(\|x-y\|, \inf_{a,b\in X}(\|x-a\| + \|f(a)-f(b)\| + \|b-y\|))$$

for some absolutely constant $c > 0$. This bound is asymptotically optimal, since no L-Lipschitz extension g can have $\|g(x)-g(y)\| > L\min(\|x-y\|, \inf_{a,b\in X}(\|x-a\| + \|f(a)-f(b)\| + \|b-y\|))$ even for a single pair of points x and y.

In some applications, one is interested in the distances $\|\tilde{f}(x) - \tilde{f}(y)\|$ between images of points $x, y \in Y$ rather than in the map $\tilde{f}$ itself. The standard Kirszbraun theorem does not provide any method of computing these distances without computing the entire map $\tilde{f}$ first. In contrast, our theorem provides a simple approximate formula for distances $\|\tilde{f}(x) - \tilde{f}(y)\|$.

2012 ACM Subject Classification Theory of computation → Computational geometry; Mathematics of computing

Keywords and phrases Kirszbraun theorem, Lipschitz map, Outer-extension, Two-sided extension

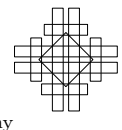
Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.13

Funding *Arturs Backurs*: Supported in part by NSF award CCF-2006806.
Konstantin Makarychev: Supported in part by NSF awards CCF-1955351 and HDR TRIPODS CCF-1934931.
Yury Makarychev: Supported in part by NSF awards CCF-1718820, CCF-1955173, and HDR TRIPODS CCF-1934843.

1 Introduction

In this paper, we prove a two-sided variant of the Kirszbraun theorem. The Kirszbraun theorem [9] is widely used in high-dimensional geometry and analysis, and has recently found applications in theoretical computer science and machine learning [1, 2, 3, 4, 5, 6, 7, 8, 10, 12, 14, 15, 16]. Recall that a function f from a subset S of Euclidean space[1] ℓ_2^n to Euclidean space ℓ_2^m is L-Lipschitz if it increases distances between points by at most a factor of L:

[1] We denote d-dimensional Euclidean space by ℓ_2^d; i.e., ℓ_2^d is $\mathbb{R}^d$ with the standard Euclidean distance.



$\|f(x) - f(y)\| \le L\|x - y\|$ for all $x, y \in X$. The Lipschitz constant $\|f\|_{Lip}$ of a function f is the minimum number L such that f is L-Lipschitz. Now, consider a subset S of Euclidean space ℓ_2^n, its superset $T \subset \ell_2^n$, and an L-Lipschitz map $f : S \to \mathbb{R}^m$. The Kirszbraun theorem states that f can be extended to a map $\tilde{f} : T \to \mathbb{R}^m$ without increasing its Lipschitz constant. That is, the theorem guarantees that $\tilde{f}$ does not increase distances by more than a factor of L. Can we guarantee that distances $\|\tilde{f}(x) - \tilde{f}(y)\|$ do not *decrease* significantly? The theorem does not provide us with any such guarantee; in fact, $\tilde{f}$ may even map distinct points to the same point and thus contract some distances infinitely.

In this paper, we prove that there exists an extension $\tilde{f}$ that does not decrease distances more than "necessary". What kind of distance contraction is necessary? Consider sets $S \subset T \subset \ell_2^n$ and an L-Lipschitz map $f : S \to \ell_2^m$. Note if f significantly contracts the distance between $a, b \in S$, then so must $\tilde{f}$, since f and $\tilde{f}$ coincide on a and b. Moreover, every L-Lipschitz extension $\tilde{f}$ must map points that are close to a and b to points close to $f(a)$ and $f(b)$, which we assumed are close to each other. More generally, consider arbitrary points $x, y \in T$, $a, b \in S$, and a cL-Lipschitz extension $\tilde{f}$ of f (where $c \ge 1$). Then, we have

$$\begin{aligned}\|\tilde{f}(x) - \tilde{f}(y)\| &\le \|\tilde{f}(x) - \tilde{f}(a)\| + \|\tilde{f}(a) - \tilde{f}(b)\| + \|\tilde{f}(b) - \tilde{f}(y)\| \\ &\le cL\|x - a\| + \|f(a) - f(b)\| + cL\|b - y\|. \quad (1)\end{aligned}$$

Since $\tilde{f}$ is cL-Lipschitz, we also have

$$\|\tilde{f}(x) - \tilde{f}(y)\| \le cL\|x - y\|. \quad (2)$$

We see that $\tilde{f}$ must satisfy (1) for all $a, b \in S$ and (2). We restate this condition in the following claim.

▷ Claim 1. Let $S \subset T \subset \ell_2^n$ and f be an L-Lipschitz map from S to ℓ_2^m. Define metric $d_{\text{ub}}(\cdot, \cdot)$ on T as follows.

$$d_{\text{ub}}(x, y) = \min\big(L\|x - y\|, \inf_{a,b \in S}(L\|x - a\| + \|f(a) - f(b)\| + L\|b - y\|)\big). \quad (3)$$

Then for every $c \ge 1$ and a cL-Lipschitz extension $\tilde{f} : T \to \ell_2^m$ of f, we have

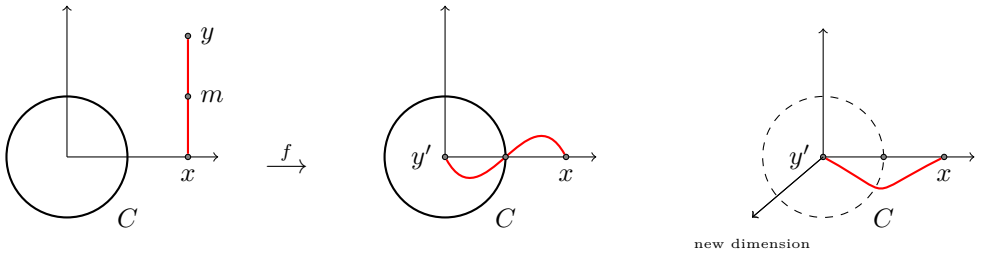
$$\|\tilde{f}(x) - \tilde{f}(y)\| \le c d_{\text{ub}}(x, y) \quad \text{for all } x, y \in T.$$

Note that $d_{\text{ub}}(\cdot, \cdot)$ satisfies all the axioms of a metric except that $d_{\text{ub}}(x, y)$ may be equal to 0 for $x \ne y$.

Our goal now is to prove a "tight" variant of the Kirszbraun theorem: there exists a Lipschitz extension $\tilde{f}$ of f such that $\|\tilde{f}(x) - \tilde{f}(y)\| \ge \Omega(d_{\text{ub}}(x, y))$ for every x and y.

However, in order to obtain such a result or, for that matter, any non-trivial result, we need to relax two conditions in the Kirszbraun theorem. We use the following example to explain what these conditions are and why we need to relax them.

▶ **Example 2.** Consider a set S that consists of a circle C around point $(0, 0) \in \ell_2^2$ and two points $x = (2, 0)$ and $y = (2, 2)$.



sets S and T extension $\tilde{f}$ in ℓ_2^2 outer extension $\tilde{f}$ in ℓ_2^3

Define f as follows: f maps each point of C to itself, x to x, and y to $y' = (0,0)$. Let $T = S \cup [x,y]$. It is immediate that f is 1-Lipschitz and $d_{\text{ub}}(u,v) > 0$ for every pair of points u and v. However, observe that the images of C and $[a,b]$ under every Lipschitz extension $\tilde{f} : T \to \ell_2^2$ necessarily intersect. Thus, every Lipschitz extension $\tilde{f}$ infinitely decreases the distance between some pair of points.

To overcome this obstacle, we consider *outer* extensions $\tilde{f}$ of f [14], which are allowed to use additional dimensions/coordinates. As standard, we denote the direct sum of Euclidean or Hilbert spaces ℓ_2^a and ℓ_2^b by $\ell_2^a \oplus \ell_2^b$, which is isometrically isomorphic to ℓ_2^{a+b}.

▶ **Definition 3.** *Let S be a non-empty subset of Euclidean space ℓ_2^n or separable Hilbert space ℓ_2^∞ and T be its superset. Consider a map $f : S \to \ell_2^m$, where m can be finite or infinite. Map $\tilde{f}$ from T to $\ell_2^m \oplus \ell_2^\Delta \simeq \ell^{m'}$ is an outer extension of $f : S \to \ell_2^m$ to $T \supset S$ (where Δ may be finite or infinite, and $m' = m + \Delta$) if it maps every point $x \in S$ to $f(x) \oplus \bar{0}$.*[2] *We will call extensions that do not use extra coordinates proper extensions.*

In Example 2, a Lipschitz *outer* extension $\tilde{f}$ can map segment $[x,y]$ to a curve that starts at x, goes above the plane ℓ_2^2, and then enters y. Map $\tilde{f}$ no longer maps distinct points to the same point. However, the Lipschitz constant of $\tilde{f}$ must be greater than 1. Indeed, let m be the midpoint between x and y. Point m is at distance 1 from each of the points x and y. If $\tilde{f}$ were 1-Lipschitz, then it would map m to a point whose distances to $x = f(x)$ and $y' = f(y)$ would not exceed 1. However, the only such point is the midpoint between x and y', which lies on C. Thus, if $\tilde{f}$ were 1-Lipschitz, it would map distinct points to the same point and thus would contract some distances infinitely. Therefore, given an L-Lipschitz map f, we will construct a $(1+\varepsilon)L$-Lipschitz outer extension $\tilde{f}$ rather than an L-Lipschitz extension. We are ready to state our main result.

▶ **Theorem 4** (Two-sided Kirszbraun Theorem)**.** *Consider $m, n \in \mathbb{Z}_{\geq 1} \cup \{\infty\}$ and $\varepsilon \in [0,1]$. Let $S \subset T$ be non-empty subsets of ℓ_2^n and f be an L-Lipschitz map from S to ℓ_2^m. Let $d_{\text{ub}}(\cdot,\cdot)$ be as defined by formula (3). There exists a $(1+\varepsilon)L$-Lipschitz outer extension $\tilde{f}$ from T to $\ell_2^m \oplus \ell_2^\Delta \simeq \ell_2^{m'}$ such that*

$$\|\tilde{f}(x) - \tilde{f}(y)\| \geq c\sqrt{\varepsilon} d_{\text{ub}}(x,y) \quad \text{for all } x, y \in T.$$

Here, c is an absolute constant. If $|T \setminus S|$ is finite, then $\Delta = O(\log |T \setminus S|)$ and $m' = m + \Delta$; if $|T \setminus S|$ is infinite, then $\Delta = m' = \infty$.

[2] That is, if m is finite, we identify ℓ_2^m with the subspace of $\ell_2^{m'}$ spanned by the first m basis vectors and allow $\tilde{f}$ to take values in $\ell_2^{m'}$.

We discuss some properties of $\tilde{f}$. It is easy to see that Claim 1 applies to both proper and outer extensions. Therefore, $\|\tilde{f}(x)-\tilde{f}(y)\| \le (1+\varepsilon)d_{\mathrm{ub}}(x,y)$. In particular, when $\varepsilon \in (0,1]$ is fixed, distances $\|\tilde{f}(x)-\tilde{f}(y)\|$ and $\Theta(d_{\mathrm{ub}}(x,y))$ are within a constant factor of each other.

$$\frac{\|\tilde{f}(x)-\tilde{f}(y)\|}{d_{\mathrm{ub}}(x,y)} \in [c\sqrt{\varepsilon}, 1+\varepsilon]. \tag{4}$$

Least possible contraction for each pair (x, y). Another property of the map $\tilde{f}$ is that it asymptotically contracts distances less than any other $(c'L)$-Lipschitz proper or outer extension g of f:

$$\|\tilde{f}(x)-\tilde{f}(y)\| \overset{\text{by Theorem 4}}{\ge} c\sqrt{\varepsilon}d_{\mathrm{ub}}(x,y) \overset{\text{by Claim 1}}{\ge} \frac{c\sqrt{\varepsilon}}{c'}\|g(x)-g(y)\| \quad \text{for all } x,y\in T.$$

Easy to compute distances. For finite subsets S and T, we can efficiently compute extensions – whose existence is guaranteed by the standard Kirszbraun theorem and Theorem 4 – using semidefinite programming (SDP).[3] However, in many proofs and applications we need to know only distances $\|\tilde{f}(x)-\tilde{f}(y)\|$ and not the map $\tilde{f}$ itself. The Kirszbraun theorem does not provide any method for obtaining the distances other than computing the entire map $\tilde{f}$, using SDP or MWU, and then directly computing $\|\tilde{f}(x)-\tilde{f}(y)\|$. In contrast, Theorem 4 provides a simple formula (4) for approximately computing all pairwise distances.

Optimal parameters

The following theorem shows that the parameters in Theorem 4 cannot be significantly improved.

▶ **Theorem 5.** *The following items hold.*

- *There exist finite sets $S \subset \ell_2^2$ and $T = S \cup \{z_1, z_2\} \subset \ell_2^2$ and a 1-Lipschitz function $f : S \to \ell_2^2$ such that the following is true. For every $\varepsilon \in (0,1]$ and a $(1+\varepsilon)$-Lipschitz extension $\tilde{f}$,*

  $$\|\tilde{f}(z_1)-\tilde{f}(z_2)\| \le O(\sqrt{\varepsilon}\, d_{\mathrm{ub}}(z_1,z_2)).$$

- *For every $m, n, N \ge 1$, there exist finite sets $S \subset T \subset \ell_2^n$ with $|T\backslash S| = N$ and a 1-Lipschitz map $f : S \to \ell_2^m$ such that the following is true. For every $c > 0$, if $\tilde{f} : \ell_2^n \to \ell_2^{m'}$ is an L-outer extension and $\|\tilde{f}(x)-\tilde{f}(y)\| \ge c\, d_{\mathrm{ub}}(x,y)$ for every $x, y \in T$, then $m' \ge c' \log_e N$ where $c' = 1/(\log_e(L/c+1))$.*
- *For every $m, n \ge 1$, there exist infinite sets $S \subset T \subset \ell_2^n$ and a 1-Lipschitz map $f : S \to \ell_2^m$ such that for every Lipschitz extension $\tilde{f} : T \to \ell_2^{m'}$ satisfying $\|\tilde{f}(x)-\tilde{f}(y)\| \ge c\, d_{\mathrm{ub}}(x,y)$ (for every $x, y \in T$ and some $c > 0$), we have $m' = \infty$.*

[3] Standard Kirszbraun extensions can be also computed using quadratically constrained quadratic programming (QCQP) or the multiplicative weight update method (MWU) [3].

1.1 Applications

Next we discuss two applications of our result.

Updating Euclidean metric

Similarity information in data sets is often encoded with Euclidean distance: two data points are similar if and only if they are close to each other (one can store either the distance itself or an embedding of points in Euclidean space). Consider the scenario where we initially have a data set X and some information about objects in X. Based on this information, we compute a Euclidean distance d_X on X. Then, we get an updated information about some subset of points Y. Using this information, we compute a new Euclidean distance d_Y on Y. Now we want to update d_X with new distances from d_Y. A natural way to do this is to first "combine" the metrics into a distance function $d_c(\cdot,\cdot)$

$$d_c = \begin{cases} d_Y(x,y), & \text{if } x,y \in Y \\ d_X(x,y), & \text{otherwise} \end{cases}. \tag{5}$$

Since d_c is not necessarily a metric, we consider the metric closure d_u of d_c (recall that by definition $d_u(x,y)$ is the length of the shortest path between x and y in the complete graph on X with edge lengths $d_c(x,y)$; see Definition 10). Metric d_u is our updated metric. We want d_u to be Euclidean (as d_X and d_Y are), but, in general, it does not have to be Euclidean.

▶ **Definition 6.** *A map f from a metric space (U,d_U) to a metric space (V,d_V) has distortion at most $D \geq 1$ if for some $c>0$ and every $x,y \in U$,*

$$cd_U(x,y) \leq d_V(f(x),f(y)) \leq cDd_U(x,y).$$

We say that a metric space (U,d_U) is D-Euclidean or that it embeds into Euclidean space with distortion at most D, if there is an embedding $f: U \to \ell_2^m$ (for some $m \in \mathbb{N} \cup \infty$) with distortion at most D.

We provide a sufficient condition when d_u is Euclidean.

▶ **Theorem 7.** *I. Consider a finite D_X-Euclidean metric space (X,d_X). Let Y be a subset of X and d_Y be a D_Y-Euclidean metric on Y. Assume that $d_Y(x,y) \leq Cd_X(x,y)$ for all $x,y \in Y$ (where $C \geq 1$). Then the updated metric d_u (defined as the closure of d_c; see (5)) is $O(CD_XD_Y)$-Euclidean.*

II. The requirement that $d_Y(x,y) \leq Cd_X(x,y)$ in item I is necessary. For every N, there exist a 1-Euclidean metric space (X,d_X) on at most N points, $Y \subset X$, and 1-Euclidean metric d_Y on Y such that every embedding of the updated metric d_u into ℓ_2 requires distortion at least $\Omega(\log N)$.

Bi-Lipschitz extension

The bi-Lipschitz constant, i.e., distortion, of a map $f\colon X \to Y$ is the minimum D such that for some $\lambda > 0$ and every $x,y \in X$, $\lambda \cdot d_X(x,y) \leq d_Y(f(x),f(y)) \leq \lambda \cdot D \cdot d_X(x,y)$. Recall that the Bi-Lipschitz map, is a map with a bounded distortion. Recently, Mahabadi, Makarychev, Makarychev, and Razenshteyn [14] proved a bi-Lipschitz variant of the Kirszbraun theorem and showed its applications to prioritized dimension reductions.

▶ **Theorem 8** (Kirszbraun theorem for bi-Lipschitz maps [14]). *Consider $S \subset T \subset \ell_2^n$ and a map $f : S \to \ell_2^m$ with distortion at most D. There exists an outer extension $\tilde{f} : T \to \ell_2^{m'}$ with distortion at most $O(D)$, where $m' = m + n$.*

This theorem follows immediately from Theorem 4 with the caveat that we do not get any bound on m' in terms of m and n (in particular, m' might be significantly greater than $m+n$). Indeed, we can assume without loss of generality that $\|f\|_{Lip} = 1$ and $\|f(x) - f(y)\| \geq \|x - y\|/D$ for all $x, y \in S$. Then $d_{\mathrm{ub}}(x, y) \geq \|x - y\|/D$. Thus for the outer extension $\tilde{f}$ from Theorem 4 (say with $\varepsilon = 1/2$), we have $\Omega(\|x - y\|/D) \leq \|\tilde{f}(x) - \tilde{f}(y)\| \leq 3/2\|x - y\|$; that is, $\tilde{f}$ has distortion at most $O(D)$.

Summary

In this paper, we prove a two-sided variant of the Kirszbraun theorem. The theorem guarantees that there is a Lipschitz outer extension $\tilde{f}$ that contracts the distance of every pair of points less than any other extension (up to a constant factor) and that has asymptotically optimal parameters. Unlike the standard Kirszbraun theorem, our theorem provides a simple approximate formula for distances $\|\tilde{f}(x) - \tilde{f}(y)\|$. Additionally, we show an application of our theorem to the Euclidean metric update problem.

Organization. In Section 2, we introduce some notation and relevant results as well as formally state the Kirszbraun theorem. In Section 3, we prove Theorem 4. In Section 4, we prove Theorem 5. Finally, in Section 5, we prove Theorem 7.

2 Preliminaries

In this paper, ℓ_2^n denotes the n-dimensional Euclidean space equipped with the standard Euclidean norm $\|\cdot\|$, when $n < \infty$; $\ell_2^\infty = \ell_2$ denotes the infinite dimensional separable Hilbert space. For $m < m'$, we identify ℓ_2^m with the m-dimensional subspace of $\ell_2^{m'}$ spanned by the first m standard basis vectors (in other words, we identify vectors $(x_1, \dots, x_m) \in \ell_2^m$ and $(x_1, \dots, x_m, 0, \dots, 0) \in \ell_2^{m'}$).

We will need the following theorem proved by Mendel and Naor [17].

▶ **Theorem 9** (Lemma 5.2 in [17], restated). *Consider space ℓ_2^n (where n is finite or infinite). For every $r > 0$, there exists a map $\psi_r : \ell_2^n \to \ell_2$ such that*[4]

$$\sqrt{\frac{e-1}{e}} \min(\|x - y\|, \sqrt{2}\, r) \leq \|\psi_r(x) - \psi_r(y)\| \leq \min(\|x - y\|, \sqrt{2}\, r)$$
$$\|\psi_r(x)\| = r$$

for all $x, y \in \ell_2^n$.

We will consider maps h from $\mathbb{R}$ to Hilbert space ℓ_2 equipped with the L_2 norm:

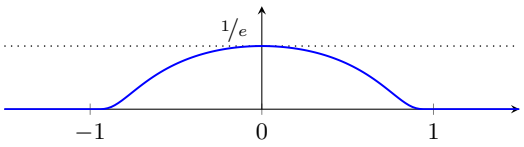
$$\|h\|_{L_2}^2 = \int_{-\infty}^{+\infty} \|h(t)\|^2 dt.$$

[4] The specific constants $\frac{e-1}{e}$ and $\sqrt{2}$ do not appear in the statement of Lemma 5.2 in [17], but can be easily deduced from the proof. Note that there is a typo on the last line of the proof of Lemma 5.2 in [17]: D should be replaced with $\sqrt{2}D$ both in the lower and upper bounds for $\|F(x) - F(y)\|_2$.

As standard, we denote the set of all such functions whose L_2-norm is finite by $L_2(\mathbb{R}, \ell_2)$. We note that Hilbert space $L_2(\mathbb{R}, \ell_2)$ is isometrically isomorphic to ℓ_2.

In the proof, we will use a "bump function" λ:

$$\lambda(t) = \begin{cases} e^{-\frac{1}{1-t^2}}, & \text{if } t \in (-1,1) \\ 0, & \text{otherwise} \end{cases}$$



We will need the following easily verifiable properties of $\lambda(t)$. Function λ is zero outside of $(-1,1)$. It is non-negative and upper bounded by $1/e$. Function λ is everywhere differentiable, and its derivative $\lambda'(t)$ is bounded by 1 in absolute value. Finally, $\lambda(t) > 1/4$ when $t \in (-1/2, 1/2)$ and $\lambda(t) > 1/7$ when $t \in (-2/3, 2/3)$.

▶ **Definition 10** (Metric Closure). *Consider a finite set of points X. Let d be a distance function on X that does not necessarily satisfy the triangle inequality (we do assume that for all $x, y \in X$: (i) $d(x,y) = 0$ if and only if $x = y$ and (ii) $d(x,y) = d(y,x)$). Denote the complete graph on X with edge lengths $d(\cdot,\cdot)$ by $K(X,d)$. The metric closure of d is the shortest path distance in $K(X,d)$.*

▶ **Theorem 11** (Kirszbraun Extension Theorem). *Consider a subset S of Euclidean space ℓ_2^n, its superset $T \subset \ell_2^n$, and an L-Lipschitz map $f : S \to \ell_2^m$. Then there exists an L-Lipschitz extension $\tilde{f} : T \to \ell_2^m$. Dimensions m and n can be finite or infinite.*

3 Two-sided Kirszbraun

In this section, we prove Theorem 4. We start with proving a simple geometric inequality (Lemma 12) and then the main lemma (Lemma 13). Theorem 4 will easily follow from Lemma 13.

Without loss of generality, we assume that set S is closed. If it is not, we let S' be the closure of S and extend f continuously to S'; then we apply the theorem to set S'. If $\varepsilon = 0$, the theorem immediately follows from the standard Kirszbraun theorem, so we assume that $\varepsilon > 0$. Let $R_x = d(x,S) = \min_{y \in S} \|x - y\|$.

▶ **Lemma 12.** *Let $u, v \in \ell_2$ be two vectors of length r and $a \geq b \geq 0$. Then*

$$\max((a-b)r, b\|u-v\|) \leq \|au - bv\| \leq (a-b)r + b\|u-v\|.$$

Proof. First, by the triangle inequality, $\|au - bv\| \leq \|au - bu\| + \|bu - bv\| = (a-b)\|u\| + b\|u-v\| = (a-b)r + b\|u-v\|$. Then $\|au - bv\| \geq \|au\| - \|bv\| = (a-b)r$. Finally, observe that $\langle u - v, u\rangle = r^2 - r^2 \cos\alpha \geq 0$, where α is the angle between u and v. Therefore, $\langle bu - bv, au - bu\rangle \geq 0$. We have $\|au - bv\|^2 = \|(bu - bv) + (au - bu)\|^2 = \|bu - bv\|^2 + \|au - bu\|^2 + 2\langle bu - bv, au - bu\rangle \geq \|bu - bv\|^2$. ◀

▶ **Lemma 13.** *There is a map $h : \ell_2^n \to L_2(\mathbb{R}, \ell_2)$ such that*

1. *h maps S to 0,*
2. *$\|h(x) - h(y)\|_{L_2} = \Theta(\min(\|x-y\|, R_x + R_y))$ for all $x, y \in \ell_2^n$.*

Proof. Let ψ be as in Theorem 9 and λ be the bump function defined in Section 2. Define $h : \ell_2^n \to L_2(\mathbb{R}, \ell_2)$ as follows:

$$h(x)(t) = \lambda(\ln R_x - t)\psi_{e^t}(x).$$

Here, we assume that $\ln 0 = -\infty$ and $\lambda(-\infty) = 0$; in other words, $h(x)(t) = 0$ if $R_x = 0$.

Let $I_x = (\ln R_x - 1, \ln R_x + 1)$ for $x \notin S$ and $I_x = \varnothing$ for $x \in S$. We have the following

- $h(x)(t) = 0$ when $t \notin I_x$ (since λ is supported on $(-1, 1)$).
- $\|h(x)(t)\| = \lambda(\ln R_x - t)\|\psi_{e^t}(x)\| = \lambda(\ln R_x - t)e^t < eR_x$ when $t \in I_x$ (since $\|\lambda\|_\infty < 1$ and $\|\psi_{e^t}(x)\| = e^t$).

We verify the first condition: If $x \in S$, then $R_x = 0$ and thus $h(x) = 0$, as required. Now we verify the second condition. Below, we assume without loss of generality that $R_x \geq R_y$.

First, we prove the desired upper bound on $\|h(x) - h(y)\|_{L_2}$ for $x, y \in \ell_2^n$. Note that if $t \notin I_x \cup I_y$, then $\|h(x)(t) - h(y)(t)\| = 0$. Further, the measure of $I_x \cup I_y$ is at most 4. For $t \in I_x \cup I_y$, $\|h(x)(t) - h(y)(t)\| \leq \|h(x)(t)\| + \|h(y)(t)\| \leq e(R_x + R_y)$. Thus,

$$\|h(x)-h(y)\|_{L_2}^2 = \int_{-\infty}^{+\infty} \|h(x)(t)-h(y)(t)\|^2 dt = \int_{I_x \cup I_y} \|h(x)(t)-h(y)(t)\|^2 dt \leq 4e^2(R_x+R_y)^2.$$

We have proved that $\|h(x) - h(y)\|_{L_2} = O(R_x + R_y)$.

Now we show that $\|h(x) - h(y)\|_{L_2} = O(\|x - y\|)$. Note that if $R_x > 2R_y$, then $\|x-y\| \geq R_x - R_y > (R_x+R_y)/3$ and therefore $\|h(x)-h(y)\|_{L_2} \leq O(R_x+R_y) \leq O(\|x-y\|)$, and we are done. So we assume that $R_x \leq 2R_y$. From Lemma 12, we get

$$\begin{aligned}\|h(x)(t) - h(y)(t)\| &= \|\lambda(\ln R_x - t)\psi_{e^t}(x) - \lambda(\ln R_y - t)\psi_{e^t}(y)\| \\ &\leq |\lambda(\ln R_x - t) - \lambda(\ln R_y - t)|e^t + \min(\lambda(\ln R_x - t), \lambda(\ln R_y - t))\|\psi_{e^t}(x) - \psi_{e^t}(y)\|.\end{aligned}$$

We upper bound the first term using the Mean Value Theorem. Using that $\|\lambda'\|_\infty \leq 1$ and $\ln a \leq a - 1$ for every $a \in \mathbb{R}$, we get that for some $\xi \in (\ln R_y - t, \ln R_x - t)$,

$$|\lambda(\ln R_x - t) - \lambda(\ln R_y - t)|e^t = |\lambda'(\xi)| \cdot |(\ln R_x - t) - (\ln R_y - t)|e^t \leq |\ln {}^{R_x}/_{R_y}|e^t \leq \frac{R_x - R_y}{R_y}e^t.$$

Now we upper bound the second term. By Theorem 9,

$$\min(\lambda(\ln R_x - t), \lambda(\ln R_y - t))\|\psi_{e^t}(x) - \psi_{e^t}(y)\| \leq O(\min(e^t, \|x - y\|)) \leq O(\|x - y\|).$$

We get

$$\begin{aligned}\|h(x) - h(y)\|_{L_2}^2 = \int_{I_x \cup I_y} \|h(x)(t) - h(y)(t)\|^2 dt &\leq 4\left(\max_{t \in I_x \cup I_y}\left(\frac{R_x - R_y}{R_y}e^t\right) + O(\|x - y\|)\right)^2 \\ &\leq O\left(\frac{R_x - R_y}{R_y} \cdot R_x + \|x - y\|\right)^2 \leq O\left(2(R_x - R_y) + \|x - y\|\right)^2.\end{aligned}$$

Since $R_x - R_y \leq \|x - y\|$, we get that $\|h(x) - h(y)\|_{L_2} = O(\|x - y\|)$.

We have proved the desired upper bound on $\|h(x) - h(y)\|_{L_2}$. Now we prove the lower bound. Consider two cases.

Case 1: $R_x \geq eR_y$. In this case, $\ln R_x \geq \ln R_y + 1$. Let $I_x^+ = (\ln R_x, \ln R_x + 1/2)$. Note that I_x^+ does not intersect I_y and thus $h(y)(t) = 0$ for $t \in I_x^+$. Also, since $\lambda(t) \geq 1/4$ on $[-1/2, 1/2]$, we have

$$\|h(y)(t)\|_2 = \lambda(\ln R_x - t)\|\psi_{e^t}(x)\| \geq {}^{e^t}/_4 \geq R_x/4$$

for $t \in I_x^+$. We have,

$$\|h(x) - h(y)\|_{L_2}^2 \geq \int_{I_x^+} \|h(x)(t) - h(y)(t)\|^2 dt = \int_{I_x^+} \|h(x)(t)\|^2 dt \geq \frac{1}{2}\left(\frac{R_x}{4}\right)^2.$$

We conclude that $\|h(x) - h(y)\|_{L_2} = \Omega(R_x + R_y)$.

Case 2: $R_x < eR_y$. In this case, $\ln R_y \le \ln R_x < \ln R_y + 1$. Let $J = (\ln R_y + 1/3, \ln R_y + 2/3)$. Then for $t \in J$, we have $\ln R_y - t \in (-2/3, -1/3)$ and $\ln R_x - t \in (-2/3, 2/3)$. Therefore, $\lambda(\ln R_x - t) \ge 1/7$ and $\lambda(\ln R_y - t) \ge 1/7$. By Lemma 12, we have for $t \in J$,

$$\begin{aligned}\|h(x)(t) - h(y)(t)\| &= \|\lambda(\ln R_x - t)\psi_{e^t}(x) - \lambda(\ln R_y - t)\psi_{e^t}(y)\| \\ &\ge \frac{1}{7}\|\psi_{e^t}(x) - \psi_{e^t}(y)\| \ge \Omega(\min(e^t, \|x-y\|)) \\ &\ge \Omega(\min(R_x + R_y, \|x-y\|))\end{aligned}$$

Using that the length of segment J is $1/3$, we get

$$\|h(x) - h(y)\|_{L_2}^2 \ge \int_J \|h(x)(t) - h(y)(t)\|^2 dt \ge \Omega(\min(R_x + R_y, \|x-y\|))^2.$$

This concludes the proof of the lower bound and the lemma. ◀

Since $L_2(\mathbb{R}, \ell_2)$ is isometrically isomorphic to ℓ_2, we get the following corollary.

▶ **Corollary 14.** *Let $T \subset \ell_2^n$. There is a map $h : T \to \ell_2^\Delta$, where $\Delta = O(\log |T \setminus S|)$ (Δ is infinite if $|T \setminus S|$ is infinite) such that*

1. *h maps S to 0,*
2. *$c \min(\|x-y\|, R_x + R_y) \le \|h(x) - h(y)\| \le \min(\|x-y\|, R_x + R_y)$ for all $x, y \in \ell_2^n$ and some absolute constant $c > 0$.*

Proof. We start with a map $h_0 : \ell_2^n \to L(\mathbb{R}, \ell_2)$ from Lemma 13. Since $L_2(\mathbb{R}, \ell_2)$ is isometrically isomorphic to ℓ_2, there is a map $h_1 : \ell_2^n \to \ell_2$ satisfying the conditions in Lemma 13. Now if $T \setminus S$ is finite, we apply Johnson-Lindenstrauss dimension reduction π_{JL} with distortion at most 3/2 to $h_1(T)$ [8]. Note that h_1 maps all points in S to 0; hence, $|h_1(T)| \le 1 + |h_1(T \setminus S)| \le 1 + |T \setminus S|$. Therefore, π_{JL} maps $h_1(T \setminus S)$ to ℓ_2^Δ with $\Delta = O(\log |T \setminus S|)$. Without loss of generality, we assume that $\pi_{JL}(0) = 0$. We get map $h_2 = \pi_{JL} \circ h_1$. If $T \setminus S$ is infinite, we simply let $h_2 = h_1$. Finally, we rescale h_2 so that the obtained map h satisfies the desired inequality:

$$c \min(\|x-y\|, R_x + R_y) \le \|h(x) - h(y)\| \le \min(\|x-y\|, R_x + R_y).$$ ◀

We are ready to prove Theorem 4. Let g be the standard Kirszbraun extension of f (Theorem 11) and h be the map provided by Corollary 14. Note that the second condition in Corollary 14 guarantees that h is 1-Lipschitz. Define $\tilde{f}(x) = g(x) \oplus (\sqrt{\varepsilon}L)h(x) \in \ell_2^m \oplus \ell_2^\Delta$.

For $x \in S$, we have $\tilde{f}(x) = f(x) \oplus 0$; thus $\tilde{f}$ is an outer extension of f. Since g is L-Lipschitz and h is 1-Lipschitz,

$$\|\tilde{f}\|_{Lip} \le \sqrt{\|g\|_{Lip}^2 + (\sqrt{\varepsilon}L)^2\|h\|_{Lip}^2} \le \sqrt{1+\varepsilon}\, L \le (1+\varepsilon)L.$$

Finally, consider arbitrary $x, y \in \ell_2^n$. If n is finite, let x' and y' be points in S closest to x and y, respectively (such points exist, since S is closed). Then $\|x - x'\| = R_x$ and $\|y - y'\| = R_y$. If n is infinite, let x' and y' be points in S such that $\|x - x'\| \le 2R_x$ and $\|y - y'\| \le 2R_y$.

Note that $\|g(x) - g(y)\| \ge \|f(x') - f(y')\| - \|g(x) - f(x')\| - \|g(y) - f(y')\| \ge \|f(x') - f(y')\| - 2L(R_x + R_y)$. We have,

$$\begin{aligned}\|\tilde{f}(x) - \tilde{f}(y)\| &= \sqrt{\|g(x) - g(y)\|^2 + \varepsilon L^2 \|h(x) - h(y)\|^2} \\ &\ge \max(\|g(x) - g(y)\|, \sqrt{\varepsilon}L\|h(x) - h(y)\|) \\ &\ge \max(\|f(x') - f(y')\| - 2L(R_x + R_y), c\sqrt{\varepsilon}L \min(\|x-y\|, R_x + R_y)).\end{aligned}$$

If $\|x-y\| \le R_x + R_y$, then we get that $\|\tilde{f}(x) - \tilde{f}(y)\| \ge c\sqrt{\varepsilon}\, L\|x-y\| \ge c\sqrt{\varepsilon}\, d_{\mathrm{ub}}(x,y)$ and we are done. So we assume now that $\|x-y\| \ge R_x + R_y$. Note that

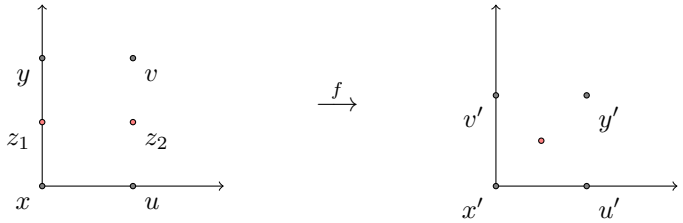
$$\begin{aligned} d_{\mathrm{ub}}(x,y) &\le \inf_{a,b\in S}(L\|x-a\| + \|f(a)-f(b)\| + L\|x-a\|) \\ &\le L\|x-x'\| + \|f(x')-f(y')\| + L\|y'-y\| \\ &\le \|f(x')-f(y')\| + 2L(R_x+R_y). \end{aligned}$$

We have

$$\begin{aligned} \|\tilde{f}(x)-\tilde{f}(y)\| &\ge \max(\|f(x')-f(y')\| - 2L(R_x+R_y), c\sqrt{\varepsilon}L(R_x+R_y)) \\ &\ge \frac{(c\sqrt{\varepsilon}/4)\cdot(\|f(x')-f(y')\| - 2L(R_x+R_y)) + 1\cdot(c\sqrt{\varepsilon}L(R_x+R_y))}{c\sqrt{\varepsilon}/4+1} \\ &= \frac{c\sqrt{\varepsilon}}{4+c\sqrt{\varepsilon}}(\|f(x')-f(y')\| + 2L(R_x+R_y)) \ge \frac{c\sqrt{\varepsilon}}{4+c\sqrt{\varepsilon}} d_{\mathrm{ub}}(x,y). \end{aligned}$$

4 Proof of Theorem 5

In this section, we prove Theorem 5. To prove item 1, consider 4 points in the plane: $x=(0,0)$, $y=(0,\sqrt{2})$, $u=(1,0)$, and $v=(1,\sqrt{2})$. Let f be the map that sends x,y,u, and v to $x'=(0,0)$, $y'=(1,1)$, $u'=(1,0)$, and $v'=(0,1)$. It is easy to see that f is 1-Lipschitz. Now let $z_1=(x+y)/2$ and $z_2=(u+v)/2$. Note that $\|x-z_1\| = \|y-z_1\| = \|u-z_2\| = \|v-z_2\| = \sqrt{2}/2$ and $d_{\mathrm{ub}}(z_1,z_2)=1$.

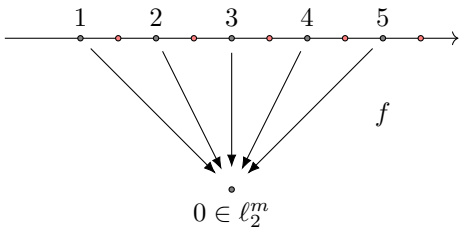


Consider a $(1+\varepsilon)$-Lipschitz proper or outer extension $\tilde{f}$ of f to $\{x,y,u,v,z_1,z_2\}$. Let $z_1' = \tilde{f}(z_1)$ and $z_2' = \tilde{f}(z_2)$. Since $\tilde{f}$ is $(1+\varepsilon)$-Lipschitz, we have $\|x'-z_1'\|^2 \le (1+\varepsilon)^2/2$ and $\|y'-z_1'\|^2 \le (1+\varepsilon)^2/2$. Hence

$$\left\|\frac{x'+y'}{2} - z_1'\right\|^2 = \frac{\|x'-z_1'\|^2 + \|y'-z_1'\|^2}{2} - \frac{\|x'-y'\|^2}{4} \le \frac{(1+\varepsilon)^2/2 + (1+\varepsilon)^2/2}{2} - \frac{1}{2} = \varepsilon + \frac{\varepsilon^2}{2}$$

Therefore, $\left\|\frac{x'+y'}{2} - z_1'\right\| = O(\sqrt{\varepsilon})$. Similarly, $\left\|\frac{u'+v'}{2} - z_2'\right\| = O(\sqrt{\varepsilon})$. Since $\frac{x'+y'}{2} = \frac{u'+v'}{2}$, we have $\|z_1'-z_2'\| = O(\sqrt{\varepsilon}) = O(\sqrt{\varepsilon}\, d_{\mathrm{ub}}(z_1',z_2'))$, as required.

Now we prove item 2. Consider sets $S = \{\kappa e_1 : 1 \le \kappa \le N\}$ and $S' = \{(\kappa + 1/2)e_1 : 1 \le \kappa \le N\}$ in ℓ_2^n (where e_1 is the first standard basis vector in ℓ_2^n). Let $T = S \cup S'$.



Consider map f that sends S to 0 in ℓ_2^m. Trivially, map f is 1-Lipschitz. We have, $d_{\text{ub}}(x,y) = \min(\|x-y\|, 1/2+0+1/2) = 1$ for every distinct $x, y \in S'$. Consider an L-Lipschitz extension $\tilde{f} : T \to \ell_2^{m'}$ such that $\|\tilde{f}(x) - \tilde{f}(y)\| \geq c d_{\text{ub}}(x,y)$. Since every point x in S' is at distance $1/2$ from some point in S, $\tilde{f}$ maps all points in S' to a ball of radius $L/2$ around 0 in $\ell_2^{m'}$. The distance between the images of every two points is at least $c d_{\text{ub}}(x,y) = c$. Therefore, $\tilde{f}(S')$ is a c-separated set in a ball of radius $L/2$ in $\ell_2^{m'}$. We apply a standard argument to bound the size of $f(S')$. Ball of radius $c/2$ around points in $\tilde{f}(S')$ are mutually disjoint and lie in a ball of radius $(L+c)/2$. Thus, the number of points in $\tilde{f}(S')$ by the ratio of the volumes of balls of radius $(L+c)/2$ and $c/2$. That is,

$$N = |\tilde{f}(S')| \leq \left(\frac{L+c}{c}\right)^{m'}.$$

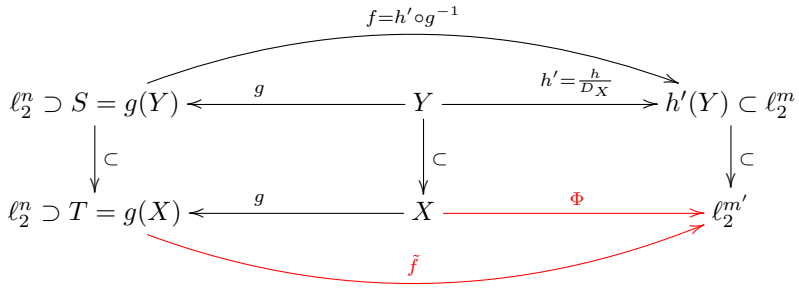
It follows that $m' \geq \frac{\log_e N}{\log_e(L/c+1)}$, as required.

Finally, observe that item 3 follows from item 2 if we let $N = \infty$.

5 Proof of Theorem 7

In this section, we prove Theorem 7.

Proof of part I. We first assume that $C = 1$ and then consider the general case when $C \geq 1$ is arbitrary. Let g and h be embeddings of (X, d_X) and (Y, d_Y) into Euclidean spaces ℓ_2^n and ℓ_2^m with distortions at most D_X and D_Y, respectively. Without loss of generality, we may assume that g and h are 1-Lipschitz and that they do not contract distances by more than D_X and D_Y, respectively. Define map h' as $h'(y) = h(y)/D_X$. Let $S = g(Y)$ and $T = g(X)$. Consider map f that sends $u \in S$ to $h'(g^{-1}(u))$.



Note that f is 1-Lipschitz, since

$$\begin{aligned}\|f(u) - f(v)\| &= \|h'(g^{-1}(u)) - h'(g^{-1}(u))\| \leq \|h'\|_{Lip} \cdot d_Y(g^{-1}(u), g^{-1}(v)) \\ &\leq \frac{\|h\|_{Lip}}{D_X} \cdot d_X(g^{-1}(u), g^{-1}(v)) \leq \frac{1}{D_X} \cdot D_X \|u - v\| = \|u - v\|.\end{aligned}$$

Consider $x, y \in X$. We show that $\frac{d_u(x,y)}{D_X D_Y} \leq d_{\text{ub}}(g(x), g(y)) \leq d_u(x,y)$ where d_u is the updated metric on X, and d_{ub} is with respect to the map f. We have,

$$\begin{aligned}d_{\text{ub}}(g(x), g(y)) &= \min(\|g(x) - g(y)\|, \inf_{a,b \in Y}(\|g(x) - g(a)\| + \|h'(a) - h'(b)\| + \|g(b) - g(y)\|)) \\ &\leq \min(d_X(x,y), \inf_{a,b \in Y}(d_X(x,a) + \frac{d_Y(a,b)}{D_X} + d_X(b,y))) \\ &\leq \min(d_c(x,y), \inf_{a,b \in Y}(d_c(x,a) + d_c(a,b) + d_c(b,y))) = d_u(x,y)\end{aligned}$$

where to get the last inequality, we need to do case analysis on the four cases given by whether $x \in Y$ or $x \notin Y$; and $y \in Y$ or $y \notin Y$. The last equality follows by considering the shortest path in d_c between x and y (of weight $d_u(x,y)$), and doing simple case analysis using the fact that $C = 1$. Then

$$\begin{aligned} d_{\mathrm{ub}}(g(x),g(y)) &= \min(\|g(x)-g(y)\|, \inf_{a,b\in Y}(\|g(x)-g(a)\| + \|h'(a)-h'(b)\| + \|g(b)-g(y)\|)) \\ &\geq \min\Big(\frac{d_X(x,y)}{D_X}, \inf_{a,b\in Y}\Big(\frac{d_X(x,a)}{D_X} + \frac{d_Y(a,b)}{D_Y D_X} + \frac{d_X(b,y)}{D_Y}\Big)\Big) \geq \frac{d_u(x,y)}{D_X D_Y}. \end{aligned}$$

By Theorem 4, there exists an 3/2-Lipschitz extension $\tilde{f} : T \to \ell_2^{m'}$ of f such that $\|\tilde{f}(u) - \tilde{f}(v)\| = \Theta(d_{\mathrm{ub}}(u,v))$ for $u, v \in T$. Consider $\Phi = \tilde{f} \circ g$, which maps X to $\ell_2^{m'}$. We obtain

$$\Omega\left(\frac{d_u(x,y)}{D_X D_Y}\right) \leq \|\Phi(x) - \Phi(y)\| \leq O(d_u(x,y)).$$

We conclude that X equipped with the updated metric d_u embeds into $\ell_2^{m'}$ with distortion at most $O(D_X D_Y)$.

To get the result for an arbitrary $C \geq 1$, we consider metric d'_Y defined by $d'_Y(x,y) = d_Y(x,y)/C$. Note that d'_Y is also D_Y-Euclidean. Additionally, $d'_Y(x,y) \leq d_X(x,y)$ for every $x, y \in Y$. Thus the updated metric d'_u for d_X and d'_Y is $O(D_X D_Y)$-Lipschitz. Finally, we note that $d'_u(x,y) \leq d_u(x,y) \leq C d'_u(x,y)$ and hence metric d_u is $O(C\, D_X D_Y)$-Euclidean. ◀

Proof of part II. Let p be the largest prime number such that $2p \leq N$. We construct an expander $G = (V, E)$ on p vertices, which is a union of two Hamiltonian paths (the paths are not necessarily disjoint). We note that any such expander will work, but we will describe one to be more specific. Let $V = \mathbb{Z}/p\mathbb{Z}$, $E_1 = \{(i, i+1) : i \in \{0, \ldots, p-2\}$, and $E'_2 = \{(i,j) : i \cdot j = 1,\ i \neq j$ where $i, j \in \mathbb{Z}/p\mathbb{Z}\}$ (where the product $i \cdot j$ is computed in $\mathbb{Z}/p\mathbb{Z}$). Observe that E'_2 is a partial matching, so we can choose a set of edges E''_2 such that $E'_2 \cup E''_2$ is a path visiting all vertices in V once. Now, $G = (V, E_1 \cup E'_2 \cup E''_2)$; P_1 and P_2 are Hamiltonian paths with edge sets E_1 and $E_2 = E'_2 \cup E''_2$, respectively. Graph $(V, E_1 \cup E'_2)$ is an expander (see [13] or Construction 4.26 in [18]) and thus so is G. Denote the shortest path distance in G by d_G.

Now, for every $i \in V$, we create two points u_i and u'_i. Let $X = \{u_i, u'_i : i \in V\}$ and $Y = \{u'_i : i \in V\}$. Consider path P_2 and one of its endpoints. Let $\pi(i)$ be the distance from vertex u_i to this endpoint along P_2. Let $\varepsilon = 1/p$. Now we define distances d_X and d_Y:

$$\begin{aligned} &d_X(u_i,u_j) = d_X(u'_i,u'_j) = |i-j| \text{ and } d_X(u_i,u'_j) = \sqrt{|i-j|^2 + \varepsilon^2}, \\ &d_Y(u'_i,u'_j) = |\pi(i) - \pi(j)|. \end{aligned}$$

Here, the value of $|i-j|$ is computed in $\mathbb{Z}$, not in $\mathbb{Z}/p\mathbb{Z}$. Observe that (X, d_X) and (Y, d_Y) embed into ℓ_2^2 and $\ell_2^1 = \mathbb{R}$ isometrically. Indeed, the map that sends u_i to $(i, 0)$ and u'_i to (i, ε) is an isometric embedding of X into the Euclidean plane; the map that sends u'_i to $\pi(i) \in \mathbb{R}$ is an isometric embedding of Y into the real line. Now consider the combined and updated metrics, d_c and d_u, on X. We have,

- $d_c(u_i,u_j) = d_X(u_i,u_j) = |i-j| \geq d_G(i,j)$;
- $d_c(u_i,u'_j) = d_X(u_i,u'_j) = \sqrt{|i-j|^2 + \varepsilon^2} > d_G(i,j)$
- $d_c(u'_i,u'_j) = d_Y(u'_i,u'_j) = |\pi(i) - \pi(j)| \geq d_G(i,j)$.

Let us derive lower and upper bounds on $d_u(u_i, u_j)$ in terms of $d_G(i, j)$. Fix some i and j. Denote the complete graph on X with edge lengths $d_c(\cdot, \cdot)$ by $K(X, d_c)$; denote the complete graph on V with edge lengths $d_G(\cdot, \cdot)$ by $K(V, d_G)$. Note that the shortest path distance in $K(V, d_G)$ equals d_G.

By definition, $d_u(u_i, u_j)$ is the shortest path distance between u_i and u_j in $K(X, d_c)$. Consider a shortest path between u_i and u_j in $K(X, d_c)$ and its "projection" to $K(V, d_G)$, which we obtain as follows. We replace each vertex on the path with a corresponding vertex in V: namely, we replace u_a and u'_a with a. We get a path from i to j in $K(V, d_G)$; its length is at least $d_G(i, j)$. Since $d_c(u_a, u_b)$, $d_c(u'_a, u_b)$, and $d_c(u'_a, u'_b)$ are all greater than or equal to $d_G(a, b)$, the length of the projected path in $K(V, d_G)$ is at most the length of the path between u_i and u_j in $K(X, d_c)$. Therefore, the shortest path distance between u_i and u_j in $K(X, d_c)$ is at least $d_G(i, j)$. We conclude that $d_u(u_i, u_j) \geq d_G(i, j)$.

Now consider a shortest path P between i and j in G. We "lift" it to $K(X, d_c)$ as follows. We replace each edge $(a, b) \in E_1$ of P with edge (u_a, u_b) in $K(X, d_c)$ and each edge $(a, b) \in E_2 \setminus E_1$ with edge (u'_a, u'_b). Note that this transformation preserves edge lengths; all of these edges have length 1. We obtain a sequence of edges in $K(X, d_c)$, which does not necessarily form a path, since it may happen that one edge ends at u_a and the next one starts at u'_a (or vice versa). We transform it to a path between u_i and u_j in $K(X, d_c)$ by adding edges of the form (u_a, u'_a) where necessary. The length of the lifted path equals the length of P, which is $d_G(i, j)$, plus the length of all the additional edges (u_a, u'_a). Each of the additional edges has length ε and there are at most p of them. We conclude that there is a path of length at most $d_G(i, j) + 1$ between u_i and u_j in $K(X, d_c)$. Thus $d_u(u_i, u_j) \leq d_G(i, j) + 1$.

We proved that $d_u(u_i, u_j) = \Theta(d_G(i, j))$. Because G is an expander, every embedding of (V, d_G) into Euclidean space requires distortion $\Omega(\log N)$ [11]. Therefore, every embedding of $(X \setminus Y, d_u)$ and, consequently, of (X, d_u) into ℓ_2 also requires distortion at least $\Omega(\log N)$. ◀

References

1. Mihai Badoiu, Kedar Dhamdhere, Anupam Gupta, Yuri Rabinovich, Harald Räcke, Ramamoorthi Ravi, and Anastasios Sidiropoulos. Approximation algorithms for low-distortion embeddings into low-dimensional spaces. In *Proceedings of the Symposium on Discrete Algorithms*, volume 5, pages 119–128, 2005.
2. Yair Bartal, Ben Recht, and Leonard J Schulman. Dimensionality reduction: beyond the Johnson–Lindenstrauss bound. In *Proceedings of the Symposium on Discrete Algorithms*, pages 868–887. SIAM, 2011.
3. Armin Biess, Aryeh Kontorovich, Yury Makarychev, and Hanan Zaichyk. Regression via Kirszbraun extension with applications to imitation learning. *CoRR*, abs/1905.11930, 2019. `arXiv:1905.11930`.
4. Yair Censor, Aviv Gibali, and Simeon Reich. Strong convergence of subgradient extragradient methods for the variational inequality problem in Hilbert space. *Optimization Methods and Software*, 26(4-5):827–845, 2011.
5. Nishant Chandgotia, Igor Pak, and Martin Tassy. Kirszbraun-type theorems for graphs. *Journal of Combinatorial Theory, Series B*, 137:10–24, 2019.
6. Timothy Chu, Gary L Miller, and Donald R Sheehy. Exact computation of a manifold metric, via lipschitz embeddings and shortest paths on a graph. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 411–425. SIAM, 2020.
7. Lee-Ad Gottlieb, Aryeh Kontorovich, and Robert Krauthgamer. Efficient regression in metric spaces via approximate Lipschitz extension. *IEEE Transactions on Information Theory*, 63(8):4838–4849, 2017.
8. William Johnson and Joram Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary Mathematics*, 26:189–206, 1984.

9 Mojzesz D. Kirszbraun. Über die zusammenziehende und Lipschitzsche Transformationen. *Fundamenta Mathematicae*, 22:77–108, 1934.

10 James R Lee and Assaf Naor. Extending Lipschitz functions via random metric partitions. *Inventiones mathematicae*, 160(1):59–95, 2005.

11 Nathan Linial, Eran London, and Yuri Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15(2):215–245, 1995.

12 Steven H Low. Analytical methods for network congestion control. *Synthesis Lectures on Communication Networks*, 10(1):1–213, 2017.

13 Alexander Lubotzky, Ralph Phillips, and Peter Sarnak. Ramanujan graphs. *Combinatorica*, 8(3):261–277, 1988.

14 Sepideh Mahabadi, Konstantin Makarychev, Yury Makarychev, and Ilya Razenshteyn. Nonlinear dimension reduction via outer bi-Lipschitz extensions. In *Proceedings of the Symposium on Theory of Computing*, pages 1088–1101, 2018.

15 Konstantin Makarychev and Yury Makarychev. A union of Euclidean metric spaces is Euclidean. *Discrete Analysis*, 14, 2016.

16 Konstantin Makarychev, Yury Makarychev, and Ilya Razenshteyn. Performance of Johnson–Lindenstrauss transform for k-means and k-medians clustering. In *Proceedings of the Symposium on Theory of Computing*, pages 1027–1038, 2019.

17 Manor Mendel and Assaf Naor. Euclidean quotients of finite metric spaces. *Advances in Mathematics*, 189(2):451–494, 2004.

18 Salil P. Vadhan. Pseudorandomness. *Foundations and Trends in Theoretical Computer Science*, 7(1–3):1–336, 2012.

Orientation Preserving Maps of the Square Grid

Imre Bárány
Alfréd Rényi Institute of Mathematics, Budapest, Hungary
Department of Mathematics, University College London, UK

Attila Pór
Department of Mathematics, Western Kentucky University, Bowling Green, KY, USA

Pavel Valtr
Department of Applied Mathematics, Faculty of Mathematics and Physics,
Charles University, Praha 1, Czech Republic

Abstract

For a finite set $A \subset \mathbb{R}^2$, a map $\varphi : A \to \mathbb{R}^2$ is *orientation preserving* if for every non-collinear triple $u, v, w \in A$ the orientation of the triangle u, v, w is the same as that of the triangle $\varphi(u), \varphi(v), \varphi(w)$. We prove that for every $n \in \mathbb{N}$ and for every $\varepsilon > 0$ there is $N = N(n, \varepsilon) \in \mathbb{N}$ such that the following holds. Assume that $\varphi : G(N) \to \mathbb{R}^2$ is an orientation preserving map where $G(N)$ is the grid $\{(i, j) \in \mathbb{Z}^2 : -N \le i, j \le N\}$. Then there is an affine transformation $\psi : \mathbb{R}^2 \to \mathbb{R}^2$ and $a \in \mathbb{Z}^2$ such that $a + G(n) \subset G(N)$ and $\|\psi \circ \varphi(z) - z\| < \varepsilon$ for every $z \in a + G(n)$. This result was previously proved in a completely different way by Nešetřil and Valtr, without obtaining any bound on N. Our proof gives $N(n, \varepsilon) = O(n^4 \varepsilon^{-2})$.

2012 ACM Subject Classification Mathematics of computing → Discrete mathematics

Keywords and phrases square grid, plane, order type

Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.14

Funding *Imre Bárány*: Partially supported by Hungarian National Research Grants No 131529, 132696, and 133819.
Pavel Valtr: Supported by the grant no. 21-32817S of the Czech Science Foundation (GAČR).

1 Introduction

This paper is about orientation preserving maps of the $n \times n$ grid. We denote by $G(N)$ the grid $\{(i, j) \in \mathbb{Z}^2 : -N \le i, j \le N\}$ and by $G^*(n)$ the grid $\{(i, j) \in \mathbb{Z}^2 : 1 \le i, j \le n\}$. A map $\varphi : G(N) \to \mathbb{R}^2$ is *orientation preserving* if for every non-collinear triple $u, v, w \in G(N)$ the orientation of the triangle u, v, w is the same as that of the triangle $\varphi(u), \varphi(v), \varphi(w)$, or with a formula

$$\operatorname{sign}\det\begin{bmatrix} u & v & w \\ 1 & 1 & 1 \end{bmatrix} = \operatorname{sign}\det\begin{bmatrix} \varphi(u) & \varphi(v) & \varphi(w) \\ 1 & 1 & 1 \end{bmatrix}.$$

We are going to show that given an orientation preserving map $\varphi : G(N) \to \mathbb{R}^2$ there is a $n \times n$ subgrid of $G(N)$ whose image under φ is very close to an affine image of the $n \times n$ grid provided N is large enough (polynomial in n and $1/\varepsilon$). Precisely we have the following result.

▶ **Theorem 1.** *For every $n \in \mathbb{N}$ and for every $\varepsilon > 0$ there is $N = N(n, \varepsilon)$ such that if $\varphi : G(N) \to \mathbb{R}^2$ is an orientation preserving map, then there is an affine transformation $\psi : \mathbb{R}^2 \to \mathbb{R}^2$ and $a \in \mathbb{Z}^2$ such that $a + G^*(n) \subset G(N)$ and for every $z \in a + G^*(n)$*

$$\|\psi \circ \varphi(z) - z\| < \varepsilon.$$

Here $N(n, \varepsilon) = O(n^4 \varepsilon^{-2})$.

Theorem 1 without the explicit bound $N(n,\varepsilon) = O(n^4\varepsilon^{-2})$ was already proved by Nešetřil and Valtr [6, Lemma 10] as the key tool for proving several Ramsey-type results (see also the paper [5] for related results). However, the proof in the paper [6] relied on repeated compactness arguments, thus it gave no upper bound on N. Our bound $N(n,\varepsilon) = O(n^4\varepsilon^{-2})$ makes ground for giving explicit bounds for Ramsey-type results given in the paper [6]; see concluding remark (1) on page 105 of the paper [6] where the lack of an explicit bound is discussed. From the (discrete and) computational geometry point of view, the most interesting consequences of our bound $N(n,\varepsilon) = O(n^4\varepsilon^{-2})$ in Theorem 1 might be those which are connected with the study of order types, as described in the next section.

We remark that the function $N(n,\varepsilon)$ in Theorem 1 satisfies the lower bound $N(n,\varepsilon) = \Omega(n^2\varepsilon^{-1})$. The example showing this is a projective map that carries the line containing one edge of the square $[-N,N]^2$ to the line at infinity.

2 Connections to order types and motivation

An *order type* of size n is an equivalence class of all n-point sets which can be mapped into each other by strongly order preserving maps, where a map $\varphi : A \to \mathbb{R}^2$ from a finite planar point set A to $\mathbb{R}^2$ is *strongly orientation preserving* if it is orientation preserving and, additionally, it maps collinear triples of A to collinear triples. If the sets of an order type are in general position then we say that the order type is *in general position*. Order types have been studied from various perspectives, for example, see the paper of Goodman and Pollack [1] for a classical result and the recent paper of Pilz and Welzl [4] for further references.

The *span* of a finite point set $A \subset \mathbb{R}^2$ is the ratio between the maximum distance in A and the minimum distance in A. Note that due to projective transformations the supremum of the spans of the sets of any fixed order type (of size at least three) is ∞. We define the *span* of an order type T as the infimum of the spans of the point sets in T. By famous results of Goodman, Pollack and Sturmfels [2] and of Kratochvíl and Matoušek [3], there are order types of size n with double exponential span.

▶ **Theorem 2.** *For $n > 1$, let $f(n)$ be the smallest real number such that, for any order type T of size n in general position and for any $\delta > 0$, there exists a set A in T having the span smaller than $f(n) + \delta$. Then there are two positive constants c_1 and c_2 such that, for any integer $n > 3$,*

$$2^{2^{c_1 n}} \le f(n) \le 2^{2^{c_2 n}}.$$

Our Theorem 1 considers subsets of sets of some order type with a small span. In particular, an immediate consequence of Theorem 1 says that some order types have the property that any set of this order type contains a rather large subset whose order type has a very small span (asymptotically as small as possible for the given size).

▶ **Theorem 3.** *For any $N \ge 2$, there is an order type T_N of size N in general position such that any set A of T_N contains a subset B of size $n = \Omega(N^{1/3})$ which is an affine transform of a set having span $O(\sqrt{n})$.*

We remark that due to a simple packing argument the span of any set (or order type) of size $n \ge 2$ is at least $\Omega(\sqrt{n})$.

Another (almost immediate) consequence of Theorem 1 says that there are order types T of arbitrary size $n \ge 2$ in general position such that any set A of order type T contains a quite large subset of points which lie, one by one, in small neighborhoods of equidistantly distributed points along some line.

▶ **Theorem 4.** *For any $N \geq 2$ and any $\varepsilon > 0$, there is an order type T_N of size N in general position such that any set A of T_N contains a subset B of size $n = \Omega(N^{1/4}\varepsilon^{1/2})$ such that for some line ℓ and for some n equally distributed points $p_1, \ldots, p_n$ on ℓ where the distance between p_i and p_{i+1} is exactly d for some fixed $d > 0$ and for each $i = 1, \ldots, n-1$, the following holds. There is exactly one point of B in the (εd)-neighborhood of p_i for each $i = 1, \ldots, n$.*

Since some of the ratios of distances among sufficiently many equidistantly distributed points on a line approximate (with any prescribed precision) l prescribed distance ratios, Theorem 4 immediately implies the following result of Nešetřil and Valtr [6, Theorem 6].

▶ **Theorem 5** (Nešetřil and Valtr [6])**.** *For any positive integer $l > 0$ and for any $l+1$ positive real numbers $\varepsilon, r_1, r_2, \ldots, r_l > 0$, there exists a (finite) order type T in general position such that any set of order type T determines $l+1$ distances $d_i, i = 0, 1, 2, \ldots, l$, such that $|\frac{d_i}{d_0} - r_i| < \varepsilon$ $(i = 1, 2, \ldots, l)$.*

3 Preparations and sketch of proof

We start with introducing basic notation and definitions. For distinct $u, v \in \mathbb{R}^2$, $L(u, v)$ denotes the line they span. The angle $\alpha(u, v)$ is defined as the angle the vector $v - u$ and the positive half of the x axis make. It is understood mod 2π. Assume $\varphi_0 : G^*(n) \to \mathbb{R}^2$ is an orientation preserving map on non-collinear triples, and it satisfies the conditions $\varphi_0(1,1) = (1,1), \varphi_0(n,1) = (n,1), \varphi_0(1,n) = (1,n)$. Suppose further that for all $u, v \in G^*(n)$ with $\alpha(u, v) \in \{0, \pi/4, \pi/2\}$

$$|\alpha(u, v) - \alpha(\varphi_0(u), \varphi_0(v))| < \gamma,$$

where $\gamma > 0$. In the last step of the proof of Theorem 1 we need the following lemma.

▶ **Lemma 6.** *Assume $\gamma = O(n^{-2})$. Then, under the above conditions for every $z \in G^*(n)$ we have*

$$\|\varphi_0(z) - z\| < 20\gamma n^2.$$

The proof is given in the last section.

Another important notion is that of a *block* of an $m \times m$ grid. The horizontal, resp. vertical blocks of $G^*(m)$ are the sets (where $i, j \in [m]$)

$$H_i = \{(1,i), (2,i), \ldots, (m,i)\} \text{ and } V_j = \{(j,1), (j,2), \ldots, (j,m)\},$$

we will call $(1,i)$ resp. (m,i) the first and last point of the block H_i, and similarly $(j,1)$ and (j,m) are the first and last points of V_j. Similarly the plus and minus diagonal blocks of $G^*(m)$ are

$$\begin{aligned} D_i^+ &= \{(x,y) \in G^*(m) : x - y = i\}, \\ D_j^- &= \{(x,y) \in G^*(m) : x + y = j\}, \end{aligned}$$

here $i = 0, \pm 1, \ldots, \pm(m-1)$ and $j = 2, \ldots, 2m$. Their first and last points are $(i+1, 1)$ and $(m, i-m)$ for D_i^+ and $(1, j-1)$ and $(j-1, 1)$ for D_j^-. Two blocks are neighbourly if they lie in consecutive parallel lattice lines.

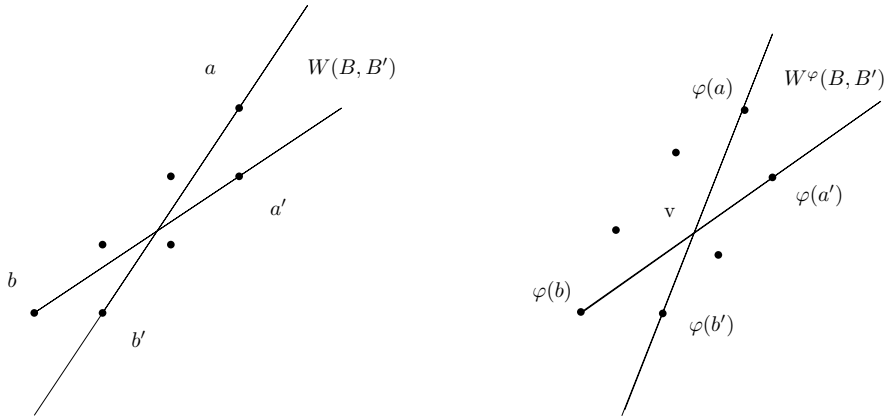


Figure 1 Neighbourly blocks and φ blocks separated.

Given an orientation preserving map $\varphi : G^*(m) \to \mathbb{R}^2$ the image $\varphi(B)$ of a block B is called a φ block. We need separation properties of blocks and φ blocks. Let B and B' be two neighbourly blocks with first and last point a, b resp. a', b'. Here $b - a$ and $b' - a'$ are parallel and point in the same direction, see Figure 1. It is clear that both $L(a, b')$ and $L(a', b)$ separate B and B'. The orientation preserving properties of φ imply that the lines

$$L_1 = L(\varphi(a), \varphi(b')) \text{ and } L_2 = L(\varphi(a'), \varphi(b))$$

also separate $\varphi(B)$ and $\varphi(B')$, or, what is the same, conv $\varphi(B)$ and conv $\varphi(B')$. The lines L_1 and L_2 define a double cone $W^\varphi(B, B')$ with apex $v = L_1 \cap L_2$ which is the double cone not containing $\varphi(B)$ and $\varphi(B')$. Similarly, let $W(B, B')$ be the double cone determined by $L(a, b')$ and $L(a', b)$, again the one not containing B and B'. The following facts are well known.

▶ **Fact 1.** *If $u \in W^\varphi(B, B')$, then $L(u, v)$ separates $\varphi(B)$ and $\varphi(B')$.*

▶ **Fact 2.** *For any point $z \in W(B, B') \cap G^*(m)$ the line $L(\varphi(z), v)$ separates $\varphi(B)$ and $\varphi(B')$.*

We say that a point $z \in \mathbb{R}^2$ is a *separator* for the horizontal blocks $H_1, \ldots, H_m$ if there are lines $L_1, \ldots, L_{m-1}$, all passing through z such that L_i separates H_i and H_{i+1} for all i. Separator points for a set of vertical and diagonal blocks, and for φ-blocks, are defined analogously.

▶ **Fact 3.** *If $z \in G(N)$ is a separator for the horizontal (or vertical, diagonal) blocks of $G^*(m)$, then so is $\varphi(z)$ for the corresponding φ blocks.*

Here is a quick sketch of the proof of Theorem 1. First we find a small subgrid, G_1, of $G(N)$. (G_1 lies in the upper halfplane and we ignore the part of $G(N)$ that is in the lower halfplane.) The points $z_h = (N, 0)$ resp. $z_- = (-N, N)$ are separators for the horizontal and minus diagonal blocks of G_1. The points v^+, v^- and w^+, w^- are separators for the vertical and plus diagonal blocks of G_1, see Figure 2. These four points are very close to the line $L(z_h, z_-)$. From the φ-image of these points we construct four collinear points that are separators for the corresponding φ blocks of G_1. A projective transformation that carries the line containing these separators to infinity can be chosen so that the horizontal resp vertical φ blocks of G_1 are separated by horizontal and vertical lines. This way we create a grid like structure. A small subgrid of G_1 can be found which satisfies the conditions of Lemma 6. The resulting map is only projective and not affine. This is to be fixed in the end.

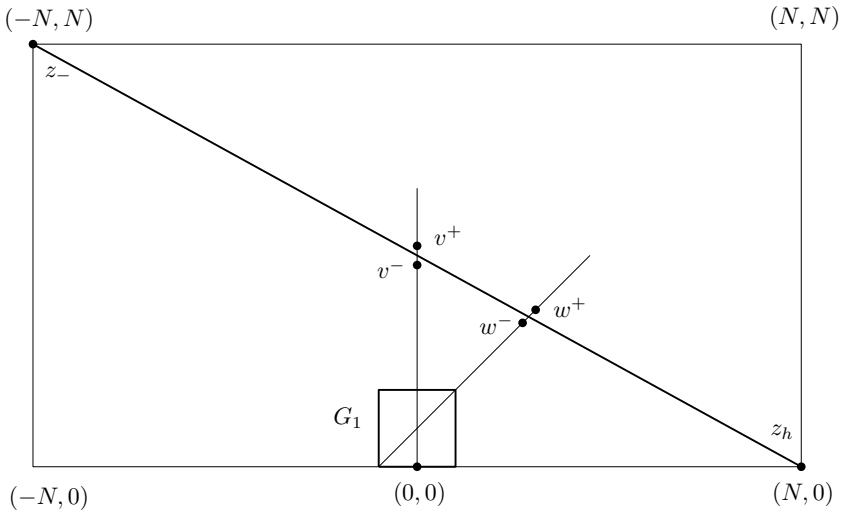


Figure 2 $G(N)$ and G_1.

4 Finding a smaller grid and a projective transformation

We set $N = 6m^2$ and define $G_1 = \mathbb{Z}^2 \bigcap([-m, m] \times [0, 2m]$, see Figure 2. The horizontal blocks of G_1 are $H_0, H_1, \ldots, H_{2m}$ and any point in $\bigcap_0^{2m-1} W(H_i, H_{i+1})$ is a separator for these blocks. An elementary calculation shows that any point $(x, 0)$ with $x > 4m^2 - m$ is a separator. In particular, $z_h = (6m^2, 0)$ is a separator. Similarly, any point $(0, y)$ with $y > 2m^2$ is a separator point for the vertical blocks of G_1. Another calculation shows that any point (x, x) resp. $(-x, x)$ with $x > m^2$ is a separator point for the plus diagonal and the minus diagonal blocks of G_1. We fix $z_- = (-6m^2, 6m^2)$ as a separator point for the minus diagonal blocks.

By Fact 3, $\varphi(z_h)$ resp. $\varphi(z_-)$ is a separator point for the horizontal and minus diagonal φ blocks of G_1. Moreover $v^+ = (3m^2 + 1, 0)$ and $v^- = (3m^2 - 1, 0)$ are both separators for the vertical blocks of G_1. Then so are $\varphi(v^+)$ and $\varphi(v^-)$ for the vertical φ blocks. These points lie on opposite sides of the line $L^\varphi = L(\varphi(z_h), \varphi(z_-))$. Consequently the intersection point, z_v, of L^φ and the segment $[\varphi(v^+), \varphi(v^-)]$ is a separator for the vertical φ blocks. Completely analogously, we find a separator $z_+ \in L^\varphi$ for the plus diagonal φ blocks of G_1. Namely, both $w^+ = (2m^2 + 1, 2m^2 + 1)$ and $w^- = (2m^2 - 1, 2m^2 - 1)$ are separators for the plus diagonal blocks of G_1, their φ-images lie on opposite sides of L^φ, so the intersection point, z_+, works again. Here is what we have established so far.

▶ **Lemma 7.** *The line L^φ contains four points $\varphi(z_h), \varphi(z_-), z_v, z_+$ that are separators for the horizontal, minus diagonal, vertical and plus diagonal φ blocks of G_1.*

Now apply a projective transformation $\psi_1 : \mathbb{R}^2 \to \mathbb{R}^2$ that maps L^φ to the line at infinity so that the horizontal resp. vertical separating lines of the corresponding φ blocks are mapped to horizontal and vertical lines of the form

$$L(b_j)^h = \{(x, y) : y = b_j\} \text{ and } L(a_i)^v = \{(x, y) : x = a_i\},$$

here $i, j \in [2m]$ and $a_1 < a_2 < \ldots < a_{2m}$ and $b_1 < b_2 < \ldots < b_{2m}$. From this point onward we only work on points of the grid that are in the lower triangular half, so that there is no reason to worry that this projective transformation might modify orientations.

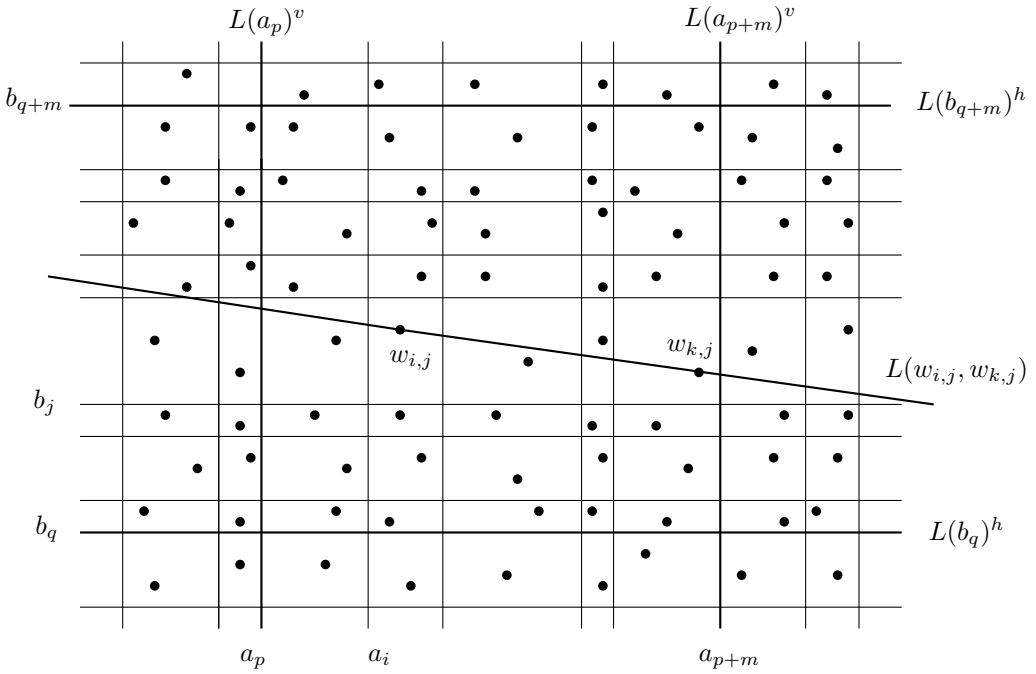


Figure 3 The grid-like structure and the line $L(w_{i,j}, w_{k,j})$.

We still have some freedom to define ψ_1 more precisely. That will come a little later. Set $\varphi_1 = \psi_1 \circ \varphi$ and note that the plus and minus diagonal φ_1 blocks of G_1 are separated by parallel lines because the corresponding separator points are at infinity. Of course, the vertical resp. horizontal φ_1 blocks are separated by the vertical and horizontal lines $L(a_i)^v$ and $L(b_j)^h$.

Observe now that we have a grid-like structure (see Figure 3): the lines $L(a_i)^v$ and $L(b_j)^h$ determine $(2m-1)^2$ rectangular cells and each such cell contains the φ_1 image of a unique point from G_1. Precisely, the cell $C(i,j)$ is just the rectangle $[a_i, a_{i+1}] \times [b_j, b_{j+1}]$. It contains the point $w_{i,j}$ which is the φ_1 image of a unique point in G_1.

Suppose that m is large, $m > 10^5$ say, and let $a_{p+5} - a_p$ resp. $b_{q+5} - b_q$ be the minimal among the numbers $a_8 - a_3, a_9 - a_4, \ldots, a_{2m-2} - a_{2m-7}$ and $b_8 - b_3, b_9 - b_4, \ldots, b_{2m-2} - b_{2m-7}$. Note that the cells in the first and last two rows and columns are not used, this "double frame" will be needed later. Here either $p < m$ or $p + 5 > m$. Similarly, either $q < m$ or $q + 5 > m$. We can assume by symmetry that $p, q < m$. We now fix ψ_1 (and so φ_1 as well) by requiring that $a_p = b_q = 0$ and $a_{p+m} = b_{q+m} = m$. It follows then that $0 < a_{p+5}, b_{q+5} \leq 5$.

We are going to show that, with φ_1 fixed this way, the angles of the plus diagonal separators are very close to $\pi/4$. A similar statement holds for the minus diagonal separators but we do not need that. We have the following lemma.

▶ **Lemma 8.** *If m is large enough, then $0 < a_{p+k+1} - a_{p+k} < 11$ and $0 < b_{q+k+1} - b_{q+k} < 11$ for all $k = -1, 0, 1, \ldots, m-1, m$.*

Proof. Let R be the rectangle $[a_p, a_{p+m}] \times [b_q, b_{q+m}]$. Define G_2 as the $m \times m$ subgrid of G_1 whose φ_1-image lies in R. Horizontal, vertical, plus and minus diagonal blocks of G_2 are defined the same way as those of G_1. Let B_i be the plus diagonal φ_1 blocks of G_2 that contains the point $w_{p,q+i}$ for $i = 0, 1, 2, 3, 4$ and let B_{-i} be the one containing $w_{p+i,q}$ for $i = 1, 2, 3, 4$.

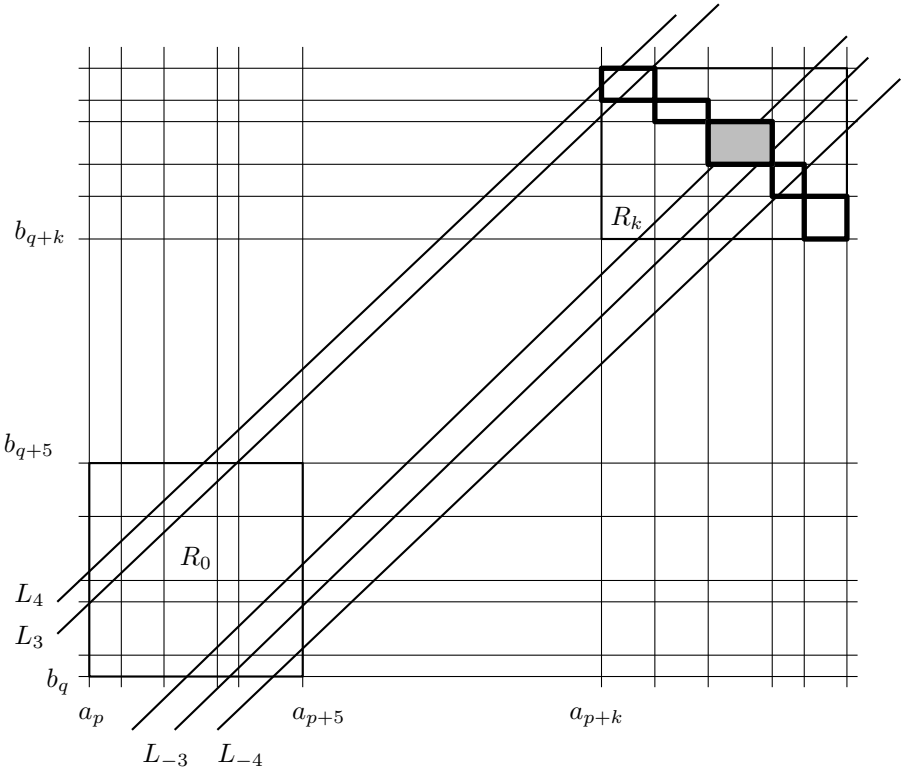


Figure 4 Only some of the lines L_i are shown, the central cell of R_k is shaded.

These diagonals are separated by parallel lines $L_4, L_3, \dots, L_{-4}$ in this order. So for instance L_1 separates B_1 and B_0, see Figure 4. Note that each such line intersects the rectangle $R_0 = [a_p, a_{p+5}] \times [b_q, b_{q+5}]$ as otherwise L_{-4} (say) would avoid it and then it cannot separate two points inside this rectangle. This implies that the distance between L_4 and L_{-4} is less then the sum of two neighbouring sides of R_0, which is at most 10.

Note further that the lines $L_{-4}, \dots, L_4$ are parallel and their slope is a positive number. Consequently the angle β these lines make with the positive half of the x axis is strictly between 0 and $\pi/2$.

Consider the rectangle $R_k = [a_{p+k}, a_{p+k+5}] \times [b_{q+k}, b_{q+k+5}]$ where $k = -2, -1, 0, \dots, m-5$. (This is the point where we use the double frame.) It contains 5^2 cells. We claim that its middle cell, $C(p+k+2, q+k+2)$, lies between the lines L_4 and L_{-4}, see Figure 4. Indeed, if it did not, then either the point (a_{p+k+3}, b_{q+k+2}) is below the line L_{-4}, or the point (a_{p+k+2}, b_{q+k+3}) is above the line L_4. In the former case the cells $C(p+k+3, q+k+1)$ and $C(p+k+4, q+k)$ also lie below the line L_{-4}. But then L_{-4} cannot separate the points $w_{p+k+3,q+k+1} \in \varphi_1(B_{-3})$ and $w_{p+k+4,q+k} \in \varphi_1(B_{-4})$, yet L_{-4} separates these two φ_1 blocks. A similar argument works when the point (a_{p+k+2}, b_{q+k+3}) is not below the line L_4.

The line L_4 intersects $L(a_p)^v$ below the point (a_p, b_{q+5}), and intersects $L(a_{p+m})^v$ above the point (a_{p+m}, b_{q+m}), so its slope is least $\frac{m-5}{m}$. Similar argument shows that the slope of the line L_{-4} is at most $\frac{m}{m-5}$. As both slopes are equal to $\tan\beta$ we have

$$\frac{m-5}{m} \le \tan\beta \le \frac{m}{m-5}. \tag{1}$$

So for m large, β is very close to $\pi/4$ and the strip between L_4 and L_{-4} (whose width is at most 10) intersects both axes in a segment of length shorter than 11. This and the fact that the central cell $C(p+k+2, q+k+2)$ lies between the lines L_4 and L_{-4} finish the proof. ◀

The next target is to show that if $w^1, w^2 \in R$ belong to the same horizontal, vertical, or plus diagonal φ_1 block, then the angle of the line $L(w^1, w^2)$ is very close to $0, \pi/2, \pi/4$. We need some preparations.

Assume that L and L' are consecutive parallel separating lines between three plus diagonal φ_1 blocks of G_2 that have points in R. Then there are four cells $C(p+k, q+h)$, $C(p+k+1, q+h)$, $C(p+k, q+h+1)$, and $C(p+k+1, q+h+1)$ so that L separates $w_{p+k,q+h+1}$ from $w_{p+k,q+h}$ and $w_{p+k+1,q+h+1}$, and L' separates $w_{p+k+1,q+h}$ from $w_{p+k,q+h}$ and $w_{p+k+1,q+h+1}$. Then both lines L and L' have to intersect the rectangle $[a_{p+k}, a_{p+k+2}] \times [b_{q+h}, b_{h+h+2}]$. The sides of this rectangle have length at most 22.

▶ **Corollary 9.** *If L and L' are consecutive parallel separating lines between three (plus or minus) diagonals φ_1 blocks of G_2, then the strip between them intersects both axes in a segment of length at most 44.*

We show next that if w^1 and w^2 belong to the same horizontal (or vertical) φ_1 block, then their line $L(w^1, w^2)$ is almost horizontal (vertical). This is quite easy now. Recall the notation $\alpha(w^1, w^2)$ for the angle of the line $L(w^1, w^2)$.

▶ **Lemma 10.** *Assume $p \le i < k \le p+m$ and $q \le j \le q+m$. Then $|\tan \alpha(w_{i,j}, w_{k,j})| < \frac{33}{m}$. Similarly $p \le i \le p+m$ and $q \le j < k \le q+m$ imply that $|\cot \alpha(w_{i,j}, w_{i,k})| < \frac{33}{m}$.*

Proof for the horizontal case. The line $L(w_{i,j}, w_{k,j})$ (see Figure 3) intersects the line $L(a_p)^v$ on the interval $[(a_p, b_{j-1}), (a_p, b_{j+2})]$, as otherwise the cell $C(p-1, j-1)$ or $C(p-1, j+1)$ from the double frame would be on the wrong side of $L(w_{i,j}, w_{k,j})$, contradicting the orientation preserving property of φ_1. Same way, the line $L(w_{i,j}, w_{k,j})$ intersects $L(a_{p+m})^v$ on the interval $[(a_{p+m}, b_{j-1}), (a_{p+m}, b_{j+2})]$. The length of both intervals is at most 33 by Lemma 8. Same proof applies in the vertical case. ◀

We want to show the analogue of Lemma 10 for plus diagonal φ_1 blocks. For that purpose we have to consider a smaller subgrid of G_2. Namely, let R' be the rectangle $[a_{p'}, a_{p'+m'}] \times [b_{q'}, b_{q'+m'}]$ anywhere near the middle of R with m' much shorter than m, $m' < \frac{m}{110}$, say. To make anywhere near more concrete choose the position of R' so that centre of R lies in R'.

▶ **Lemma 11.** *Assume $w^1, w^2 \in R'$ belong to the same plus diagonal φ_1 block of G_2, and $\alpha(w^1, w^2)$ differs from $\pi/4$ by δ. Then $|\tan \delta| < \frac{K}{m}$ where K is a constant, for instance $K = 400$ will do.*

Proof. Let B_0 be the plus diagonal φ_1 block containing w^1, w^2, and let $B_{-2}, B_{-1}, B_0, B_1, B_2$ the neighbouring diagonal φ_1 blocks, with parallel separating lines L_{-2}, L_{-1}, L_1, L_2. The strip between the lines L_{-2} and L_2 intersects the lines $L(a_{p+m/10})^v$ and $L(a_{p+9m/10})^v$ in two segments that lie in the rectangle R, and have length at most $3 \cdot 44$ with 44 coming from Corollary 9. Same way as in the proof of Lemma 10, the line $L(w^1, w^2)$ has to intersect these two segments. A straightforward computation, using this fact and (1), finishes the proof. ◀

▶ Remark. In this proof we use the line $L(a_{p+m/10})^v$ (instead of $L(a_p)^v$) because its intersection with the strip between L_{-2} and L_2 should lie inside R. The same reason explains the line $L(a_{p+9m/10})^v$.

5 Finding an even smaller subgrid

We set $m = Cn^2\varepsilon^{-1}$ where $C > 0$ will be specified later. Let G_3 be the subgrid of G_2, a translate of the set of grid points in $[0, n]^2$ such that φ_1 maps the bottom left corner of G_3 to the point $(a_{p'}, b_{q'})$. Note that $n < m'$, in fact much smaller. The set $\varphi_1(G_3)$ is contained in the rectangle $R^* = [a_{p'}, a_{p'+n}] \times [b_{q'}, b_{q'+n}]$ whose sides are shorter than $11n$.

We define an affine map $\psi_2 : \mathbb{R}^2 \to \mathbb{R}^2$ by requiring $\psi_2(w_{p',q'}) = (0, 0), \psi_2(w_{p'+n,q'}) = (n, 0)$ and $\psi_2(w_{p',q'+n}) = (0, n)$. Then $\varphi_2 = \psi_2 \circ \varphi_1$ is well-defined on G_3. The map ψ_2 hardly changes any direction. More precisely, Lemmas 10 and 11 imply the following.

▶ **Fact 4.** *If z^1, z^2 belong to the same horizontal, plus diagonal, vertical block of G_3, then $\alpha(\varphi_2(z^1), \varphi_2(z^2))$ deviates from $0, \pi/4, \pi/2$ by at most 2δ where $|\tan\delta| < K/m$.*

The conditions of Lemma 6 are satisfied with $\gamma = 2 \arctan K/m$. Thus its conclusion holds: for every $z \in G_3$

$$\|\varphi_2(z) - z\| < 20\gamma n^2 \le 40n^2 \arctan\frac{2K\varepsilon}{Cn^2} < \frac{80K}{C}\varepsilon.$$

We are almost finished, except that $\psi_3 = \psi_2 \circ \psi_1$ is not an affine but a projective transformation. It is of the form

$$\psi_3(x) = \frac{Ax}{\ell(x)}$$

where A is an orientation preserving affine map, and $\ell(x)$ is the equation of the line L^φ, normalized so that $\ell(x)$ is the signed distance of x from the line L^φ. This line goes to infinity under ψ_1 and is disjoint from R and then is far from R^*; let d denote their distance. As $n, m' < \frac{m}{110}$, the side length of R^* is at most $11m' < \frac{m}{10}$. Since R^* is in the middle of R, this implies that $d > \frac{4m}{10}$. The diameter of R^* is at most $11n\sqrt{2}$, very small compared to m. Then for every $x \in R^*$, $d \le \ell(x) \le d + 9n\sqrt{2}$. Consequently, using $m = Cn^2\varepsilon^{-1}$

$$1 \le \frac{\ell(x)}{d} \le 1 + \frac{11n\sqrt{2}}{.4Cn^2\varepsilon^{-1}} < 1 + \frac{40\varepsilon}{Cn}.$$

The map $\psi(x) = Ax/d$ is affine and satisfies

$$\|\psi(z) - z\| \le \|\psi(z) - \psi_3(z)\| + \|\psi_3(z) - z\|.$$

Here $Az/\ell(z)$ is inside the square $[0, n]^2$ or very close to it, so its norm is at most $2n$. Then

$$\|\psi(z) - \psi_3(z)\| = \left\|\frac{Az}{\ell(z)}\right\| \frac{\ell(z) - d}{d} \le 2n\frac{40\varepsilon}{Cn} = \frac{80\varepsilon}{C}.$$

Thus

$$\|\psi(z) - z\| < \frac{80\varepsilon}{C} + \frac{80K\varepsilon}{C} < \varepsilon,$$

when C is chosen larger than $80K + 80$. ◀

6 Proof of Lemma 6

Proof. Consider the quadrilateral $Q = \text{conv}\,\{A, B, C, D\}$ as in Figure 5. Assume that

$$\begin{aligned} |\alpha(A,B)|, |\alpha(D,C)| < \gamma, |\alpha(A,C) - \pi/4| &< \gamma \\ |\alpha(A,D) - \pi/2|, |\alpha(B,C) - \pi/2| &< \gamma. \end{aligned}$$

The sine theorem shows that, with the notation of Figure 5,

$$\frac{d}{\sqrt{2}}(1 - \tan 2\gamma) < a, a', b, b' < \frac{d}{\sqrt{2}}(1 + \tan 2\gamma).$$

Setting $M = \frac{1+\tan 2\gamma}{1-\tan 2\gamma}$ it follows that

$$M^{-1} < \frac{a}{a'}, \frac{b}{a}, \frac{b'}{a}, \frac{a}{b'} < M$$

We are going to use these inequalities in the quadrilaterals whose vertices are $\varphi_0(i,j), \varphi_0(i+1,j), \varphi_0(i,j+1), \varphi_0(i+1,j+1)$. We define $a_{i,j} = \varphi_0(i+1,j) - \varphi_0(i,j)$ and $b_{i,j} = \varphi_0(i,j+1) - \varphi_0(i,j)$. We write a^x, a^y for the x and y components of the vector $a \in M^2$.

The above inequalities show that in the triangle with sides $a_{i-1,1}$ and $b_{i,1}$ (see Figure 5), and in the triangle with sides $b_{i,1}$ and $a_{i,1}$

$$M^{-1} < \frac{\|b_{i,1}\|}{\|a_{i-1,1}\|} < M \text{ and } M^{-1} < \frac{\|a_{i,1}\|}{\|b_{i,1}\|} < M.$$

Consequently

$$M^{-2} < \frac{\|a_{i,1}\|}{\|a_{i-1,1}\|} < M^2 \text{ and so } \max \|a_{i,1}\| \le \min \|a_{i,1}\| M^{2(n-1)}.$$

As $a^x_{i,1} > 0$ follows from the conditions, and $a^x_{i,1} \ge \|a_{i,1}\| \cos\gamma$, we have

$$\frac{\max a^x_{i,1}}{\min a^x_{i,1}} \le \frac{\|\max a_{i,j}\|}{\|\min a_{i,j}\| \cos\gamma} < \frac{M^{2(n-1)}}{\cos\gamma} =: 1 + \Delta.$$

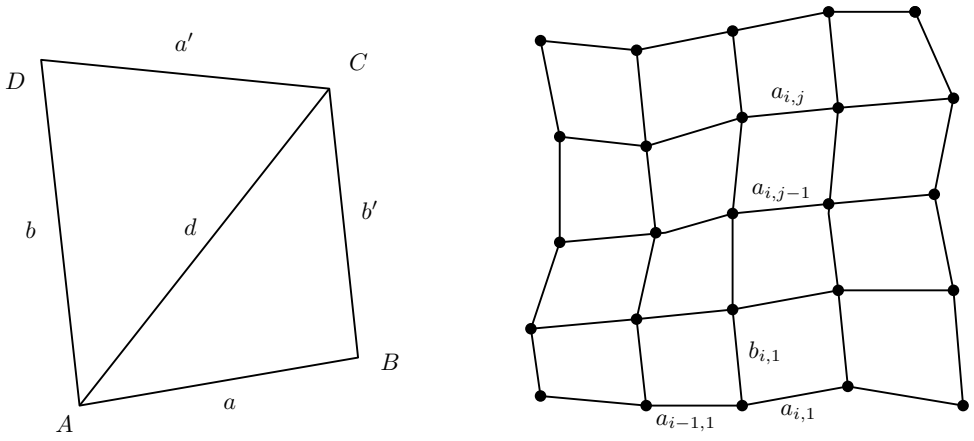


Figure 5 The quadrilateral Q and a piece of the φ_0-grid.

The average of the $a^x_{i,1}$ for $i \in [n-1]$ is 1 because $\sum_1^{n-1} a^x_{i,1} = n-1$, so $\min a^x_{i,1} \le 1 \le \max a^x_{i,x}$ implying that $\max a^x_{i,x} < (1+\Delta)\min a^x_{i,1} \le 1+\Delta$, and $\min a^x_{i,1} > \max a^x_{i,1}/(1+\Delta) \ge 1-\Delta$. Consequently

$$|a^x_{i,1} - 1| \le \Delta \text{ for all } i \in [n-1].$$

We need to estimate Δ:

$$\begin{aligned}
\Delta &= \frac{M^{2(n-1)}}{\cos\gamma} - 1 = \left(\frac{1+\tan 2\gamma}{1-\tan 2\gamma}\right)^{2n} \frac{1}{\cos\gamma} - 1 \\
&\le \left(1 + \frac{2\tan 2\gamma}{1-\tan 2\gamma}\right)^{2(n-1)} (1-\gamma^2)^{-1/2} - 1 \\
&\le \exp\left\{2(n-1)\frac{2\tan 2\gamma}{1-\tan 2\gamma}\right\}(1+\gamma) - 1 \le 10n\gamma.
\end{aligned}$$

The last inequality follows from $e^t \le 1 + 1.1t$ which is true if $t > 0$ is small enough. This is the case as $t = 2(n-1)\frac{2\tan 2\gamma}{1-\tan 2\gamma} \approx 8n\gamma$. Consequently

$$|a^x_{i,1} - 1| \le 10n\gamma \text{ and similarly } |b^y_{1,j} - 1| \le 10n\gamma. \tag{2}$$

In the quadrilateral with sides $a_{i-1,j}$ and $a_{i,j}$ (see Figure 5 again) we have, the same way as in Q above, that

$$\frac{\|a_{i,j}\|}{\|a_{i,j-1}\|} < M \text{ and so } \|a_{i,j}\| \le M^{n-1}\|a_{i,1}\|.$$

Since M^{n-1} is only slightly larger than 1 and $|a^y_{i,j}| \le a^x_{i,j}\sin\gamma$ we have

$$|a^y_{i,1}| \le 2\gamma \text{ and similarly } |b^y_{1,j}| \le 2\gamma. \tag{3}$$

Finally we estimate the difference $\varphi_0(i,j) - (i,j)$. The absolute value of the x component of this vector is

$$\begin{aligned}
&= |1 + a^x_{1,1} + \ldots + a^x_{i-1,1} + b^x_{i,1} + \ldots b^x_{i,j-1} - i| \\
&\le |a^x_{1,1} - 1| + \ldots + |a^x_{i-1,1} - 1| + |b^x_{i,1}| + \ldots + |b^x_{i,j-1}| \\
&\le (i-1)10n\gamma + (j-1)2\gamma \le (n-1)(10n\gamma + 2\gamma) < 10n^2\gamma,
\end{aligned}$$

where we used (2) and (3). Estimating the y component of the vector $\varphi_0(i,j) - (i,j)$ is similar but starts with writing this vector as

$$b_{1,1} + \ldots + b_{1,j-1} + a_{j,1} + \ldots a_{j,i-1}.$$ ◀

References

1 Jacob E. Goodman and Richard Pollack. The complexity of point configurations. *Discret. Appl. Math.*, 31(2):167–180, 1991. doi:10.1016/0166-218X(91)90068-8.

2 Jacob E. Goodman, Richard Pollack, and Bernd Sturmfels. The intrinsic spread of a configuration in $\mathbb{R}^d$. *Journal of the American Mathematical Society*, 3(3):639–651, 1990. URL: http://www.jstor.org/stable/1990931.

3 Jan Kratochvíl and Jiří Matoušek. Intersection graphs of segments. *J. Comb. Theory, Ser. B*, 62(2):289–315, 1994. doi:10.1006/jctb.1994.1071.

4 Alexander Pilz and Emo Welzl. Order on order types. *Discret. Comput. Geom.*, 59(4):886–922, 2018. doi:10.1007/s00454-017-9912-9.

5 Jaroslav Nešetřil and Pavel Valtr. A Ramsey-type theorem in the plane. *Comb. Probab. Comput.*, 3:127–135, 1994. doi:10.1017/S0963548300001024.

6 Jaroslav Nešetřil and Pavel Valtr. A Ramsey property of order types. *J. Comb. Theory, Ser. A*, 81(1):88–107, 1998. doi:10.1006/jcta.1997.2820.

Light Euclidean Steiner Spanners in the Plane

Sujoy Bhore
Université Libre de Bruxelles, Belgium

Csaba D. Tóth
California State University Northridge, Los Angeles, CA, USA
Tufts University, Medford, MA, USA

Abstract

Lightness is a fundamental parameter for Euclidean spanners; it is the ratio of the spanner weight to the weight of the minimum spanning tree of a finite set of points in $\mathbb{R}^d$. In a recent breakthrough, Le and Solomon (2019) established the precise dependencies on $\varepsilon > 0$ and $d \in \mathbb{N}$ of the minimum lightness of a $(1+\varepsilon)$-spanner, and observed that additional Steiner points can substantially improve the lightness. Le and Solomon (2020) constructed Steiner $(1+\varepsilon)$-spanners of lightness $O(\varepsilon^{-1} \log \Delta)$ in the plane, where $\Delta \geq \Omega(\sqrt{n})$ is the *spread* of the point set, defined as the ratio between the maximum and minimum distance between a pair of points. They also constructed spanners of lightness $\tilde{O}(\varepsilon^{-(d+1)/2})$ in dimensions $d \geq 3$. Recently, Bhore and Tóth (2020) established a lower bound of $\Omega(\varepsilon^{-d/2})$ for the lightness of Steiner $(1+\varepsilon)$-spanners in $\mathbb{R}^d$, for $d \geq 2$. The central open problem in this area is to close the gap between the lower and upper bounds in all dimensions $d \geq 2$.

In this work, we show that for every finite set of points in the plane and every $\varepsilon > 0$, there exists a Euclidean Steiner $(1+\varepsilon)$-spanner of lightness $O(\varepsilon^{-1})$; this matches the lower bound for $d = 2$. We generalize the notion of shallow light trees, which may be of independent interest, and use directional spanners and a modified window partitioning scheme to achieve a tight weight analysis.

2012 ACM Subject Classification Mathematics of computing → Approximation algorithms; Mathematics of computing → Paths and connectivity problems; Theory of computation → Computational geometry

Keywords and phrases Geometric spanner, lightness, minimum weight

Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.15

Related Version *Full Version*: https://arxiv.org/abs/2012.02216

Funding *Sujoy Bhore*: Research on this paper was supported by the Fonds de la Recherche Scientifique-FNRS under Grant no MISU F 6001.
Csaba D. Tóth: Research on this paper was partially supported by the NSF award DMS-1800734.

1 Introduction

Given an edge-weighted graph G, a spanner is a subgraph H of G that preserves the length of the shortest paths in G up to some amount of multiplicative or additive distortion. Formally, a subgraph H of a given edge-weighted graph G is a *t-spanner*, for some $t \geq 1$, if for every $pq \in \binom{V(G)}{2}$ we have $d_H(p,q) \leq t \cdot d_G(p,q)$, where $d_G(p,q)$ denotes the length of the shortest path in G. The parameter t is called the *stretch factor* of the spanner. Graph spanners were introduced by Peleg and Schäffer [40], and since then it has turned out to be a fundamental graph structure with numerous applications in the field of distributed systems and communication, distributed queuing protocol, compact routing schemes, etc.; see [19, 29, 41, 42]. For edge-weighted graphs, a natural parameter is the *lightness* of a spanner, that is associated with the total weight of the spanner. The *lightness* of a spanner H of an input graph G, is the ratio $w(H)/w(\text{MST})$ between the total weight of H and the weight of a minimum spanning tree (MST) of G. Note that, since a spanner H is a connected graph, the trivial lower bound for lightness is 1.

In geometric settings, a t-spanner for a finite set S of points in $\mathbb{R}^d$ is a subgraph of the underlying complete graph $G = (S, \binom{S}{2})$, that preserves the pairwise Euclidean distances between points in S to within a factor of t, that is the *stretch factor*. The edge weights of G are the Euclidean distances between the vertices. Chew [14, 15] initiated the study of Euclidean spanners in 1986, and showed that for a set of n points in $\mathbb{R}^2$, there exists a spanner with $O(n)$ edges and constant stretch factor. Since then a large body of research has been devoted to Euclidean spanners due to its many applications across domains, such as, topology control in wireless networks [45], efficient regression in metric spaces [26], approximate distance oracles [28], and others. Moreover, Rao and Smith [43] showed the relevance of Euclidean spanners in the context of other fundamental geometric NP-hard problems, e.g., Euclidean traveling salesman problem and Euclidean minimum Steiner tree problem. Many different spanner construction approaches have been developed for Euclidean spanners over the years, that each found further applications in geometric optimization, such as spanners based on well-separated pair decomposition (WSPD) [11, 27], skip-lists [3], path-greedy and gap-greedy approaches [1, 4], locality-sensitive orderings [12], and more. We refer to the book by Narasimhan and Smid [39] and the survey of Bose and Smid [10] for a summary of results and techniques on Euclidean spanners up to 2013.

Lightness and sparsity are two natural parameters for Euclidean spanners. For a set S of points in $\mathbb{R}^d$, the lightness is the ratio of the spanner weight (i.e., the sum of all edge weights) to the weight of the Euclidean minimum spanning tree $\mathrm{MST}(S)$. Then, the *sparsity* of a spanner on S is the ratio of its size to the size of a spanning tree. Classical results show that when the dimension $d \in \mathbb{N}$ and $\varepsilon > 0$ are constant, every set S of n points in d-space admits an $(1+\varepsilon)$-spanners with $O(n)$ edges and weight proportional to that of the Euclidean MST of S. We refer to a series of spanners constructions for a comprehensible understanding of sparse spanners [15, 16, 30, 31, 44, 49].

Of particular interest, we elaborate on the *lightness* aspect of Euclidean spanners. Das et al. [17] showed that the *greedy-spanner* (cf. [1]) has constant lightness in $\mathbb{R}^3$. This was generalized later to $\mathbb{R}^d$, for all $d \in \mathbb{N}$, by Das et al. [18]. However the dependencies on ε and d have not been addressed. Rao and Smith [43] showed that the greedy spanner has lightness $\varepsilon^{-O(d)}$ in $\mathbb{R}^d$ for every constant d, and asked what is the best possible constant in the exponent. A complete proof for the existance of a $(1+\varepsilon)$-spanner with lightness $O(\varepsilon^{-2d})$ is in the book on geometric spanners [39]. Gao et al. [24] considered the spanners in doubling metrics, and showed that every finite set of n points in doubling dimension d has a spanner of sparsity $\varepsilon^{-O(d)}$. In 2015, Gottlieb [25] showed that a metric of doubling dimension d has a spanner of lightness $(d/\varepsilon)^{O(d)}$, which improved the $O(\log n)$ lightness bound of Smid [46]. Recently, Borradaile et al. [9] showed that the greedy $(1+\varepsilon)$-spanner of a finite metric space of doubling dimension d has lightness $\varepsilon^{-O(d)}$. In [33], Le and Solomon established the precise dependencies of ε in the *lightness* and *sparsity* bounds of Euclidean $(1+\varepsilon)$-spanners. They constructed, for every $\varepsilon > 0$ and constant $d \in \mathbb{N}$, a set S of n points in $\mathbb{R}^d$ for which any $(1+\varepsilon)$-spanner must have lightness $\Omega(\varepsilon^{-d})$ and sparsity $\Omega(\varepsilon^{-d+1})$, whenever $\varepsilon = \Omega(n^{-1/(d-1)})$. Moreover, they showed that the greedy $(1+\varepsilon)$-spanner in $\mathbb{R}^d$ has lightness $O(\varepsilon^{-d} \log \varepsilon^{-1})$.

Steiner Spanners. *Steiner points* are additional vertices in a network that are not part of the input, and a t-spanner must achieve stretch factor t only between pairs of the input points in S. Le and Solomon [33] observed that it is possible to use Steiner points to bypass the lower bounds and substantially improve the bounds for *lightness* and *sparsity* of Euclidean $(1+\varepsilon)$-spanners. For minimum sparsity, they gave an upper bound of $O(\varepsilon^{(1-d)/2})$ for d-space

and a lower bound of $\Omega(\varepsilon^{-1/2}/\log\varepsilon^{-1})$. For minimum lightness, they gave a lower bound of $\Omega(\varepsilon^{-1}/\log\varepsilon^{-1})$, for points in the plane ($d = 2$) [33]. In a subsequent work [34], they have constructed Steiner $(1+\varepsilon)$-spanners of lightness $O(\varepsilon^{-1}\log\Delta)$ in the plane, where Δ is the *spread* of the point set, defined as the ratio between the maximum and minimum distance between a pair of points. In particular, $\log\Delta \in \Omega(\log n)$ in doubling metrics.

Recently, Bhore and Tóth [7] established a lower bound of $\Omega(\varepsilon^{-d/2})$ for the lightness of Steiner $(1+\varepsilon)$-spanners in Euclidean d-space for all $d \geq 2$. Moreover, for points in the plane, they established an upper bound of $O(\varepsilon^{-1}\log n)$. In [35], Le and Solomon constructed spanners of lightness $\tilde{O}(\varepsilon^{-(d+1)/2})$ in dimensions $d \geq 3$, nearly matching the lower bound $\Omega(\varepsilon^{-d/2})$, for $d \geq 3$. The central open problem in this area is to close the gap between the lower and upper bounds of lightness, in all dimensions $d \geq 2$.

▶ **Question 1.** *Do there exist Euclidean Steiner $(1+\varepsilon)$-spanners for a finite set of points in $\mathbb{R}^d$, of lightness $O(\varepsilon^{-d/2})$, for any $d \geq 2$?*

Bounding the *lightness* of Euclidean spanners is often harder than bounding the *sparsity*, as also noted by Le and Solomon [34]. Several works portrayed the difficulties of constructing light spanners in Euclidean spaces, doubling metrics, as well as on other weighted graphs; see [1, 9, 13, 22, 25, 33, 46, 18, 43]. A delicate aspect of the problem is to find suitable locations for *Steiner points*. Recent results on Steiner spanners [7, 33, 34, 35] suggest that highly nontrivial insights are required to argue the upper bounds for Steiner spanners, and they tend to be even more intricate than their non-Steiner counterpart.

Related Previous Work. Steiner points were used in several occasions to improve the overall weight of a network. Previously, Elkin and Solomon [23] and Solomon [47] showed that Steiner points can improve the weight of the network in the single-source setting. In particular, they introduced the so-called *shallow-light trees* (SLT), that is a single-source spanning tree that concurrently approximates a shortest-path tree (between the source and all other points) and a minimum spanning tree (for the total weight). They proved that Steiner points help to obtain exponential improvement on the lightness of SLTs in a general metric space [23], and quadratic improvement on the lightness in Euclidean spaces [47].

Our Contribution. In this work, we show that for every finite set of points in the plane and every $\varepsilon > 0$, there exists a Euclidean Steiner $(1+\varepsilon)$-spanner of lightness $O(\varepsilon^{-1})$ (Theorem 2). This matches the lower bound for $d = 2$, and thereby closes the gap between lower and upper bounds of lightness for Euclidean $(1+\varepsilon)$-spanners in $\mathbb{R}^2$.

On the one hand, without Steiner points, the greedy spanner in Euclidean plane has lightness $\tilde{O}(\varepsilon^{-2})$, which is the best possible up to lower-order terms [33]. On the other hand, with Steiner points, recent constructions achieved linear dependence on ε^{-1}, while loosing the independence from n; see [7, 34]. Our result is the first that constructs Steiner spanners with sub-quadratic dependence on ε^{-1} without any dependence on n or any assumption on the point set, in fact our result achieves the optimal dependence on ε.

▶ **Theorem 2.** *For every finite point sets $S \subset \mathbb{R}^2$ and $\varepsilon > 0$, there exists a Euclidean Steiner $(1+\varepsilon)$-spanner of weight $O(\frac{1}{\varepsilon}\,\|MST(S)\|)$.*

Outline. We review previous results on angle-bounded paths, SLTs, and window partitions that we use in our construction (Section 2). The tight bound in Theorem 2 relies on three new ideas, which may be of independent interest: First, we generalize Solomon's SLTs to points on a staircase path (Section 3). Second, we reduce the proof of Theorem 2 to the

construction of "directional" spanners, in each of $\Theta(\varepsilon^{-1/2})$ directions, where it is enough to establish the stretch factor $1+\varepsilon$ for point pairs $s,t \in S$ where $\mathrm{dir}(st)$ is in an interval of size $\sqrt{\varepsilon}$ (Section 4). Combining the first two ideas, we show how to construct light directional spanners for points on a staircase path (Section 5). In each direction, we start with a rectilinear MST of S, and augment it into a directional spanner. We modify the classical window partition of a rectilinear polygon into histograms by replacing vertical edges with angle-bounded paths; this is the final piece of the puzzle. Near-vertical paths (with slopes $\pm\varepsilon^{-1/2}$) allow sufficient flexibility to reduce the weight of a histogram subdivision, and we can construct directional $(1+\varepsilon)$-spanners for each face of such a subdivision (Section 6).

2 Preliminaries

The *direction* of a line segment ab in the plane, denoted $\mathrm{dir}(ab)$, is the minimum counterclockwise angle $\alpha \in [0,\pi)$ that rotates the x-axis to be parallel to ab. The set of possible directions $[0,\pi)$ is homeomorphic to the unit circle $\mathbb{S}^1$, and an interval (α,β) of directions corresponds to the counterclockwise arc of $\mathbb{S}^1$ from $\alpha \pmod\pi$ to $\beta \pmod\pi$.

Angle-Bounded Paths. For $\delta \in (0,\pi/2]$, a polygonal path $(v_0,\ldots,v_m)$ is $(\theta\pm\delta)$-*angle-bounded* if the direction of every segment $v_{i-1}v_i$ is in the interval $[\theta-\delta,\theta+\delta]$; see Fig. 1(a). Borradaile and Eppstein [8, Lemma 5] observed that the weight of a $(\theta\pm\delta)$-angle-bounded st-path is at most $(1+O(\delta^2))\|st\|$. We prove this observation in a more precise form. The quadratic growth rate in δ is due to the Taylor estimate $\sec(x) = \frac{1}{\cos(x)} \le 1+x^2$ for $x \le \frac{\pi}{4}$.

▶ **Lemma 3.** *Let $a,b \in \mathbb{R}^2$ and let $P = v_0v_1\ldots v_m$ be an ab-path such that P is monotonic in direction $\overrightarrow{ab}$ and $|\mathrm{dir}(v_{i-1}v_i) - \mathrm{dir}(ab)| \le \delta \le \frac{\pi}{4}$, for $i = 1,\ldots,m$. Then $\|P\| \le (1+\delta^2)\|ab\|$.*

Proof. For $i = 0,\ldots,m$, let u_i be the orthogonal projection of v_i to the line ab, and let $\alpha_i = \mathrm{dir}(v_{i-1}v_i) - \mathrm{dir}(ab)$. Then $\|ab\| = \sum_{i=1}^m \|u_{i-1}u_i\| = \sum_{i=1}^m \|v_{i-1}v_i\|\sec\angle_i \le \|P\|\sec\delta \le (1+\delta^2)\|P\|$, as claimed. ◀

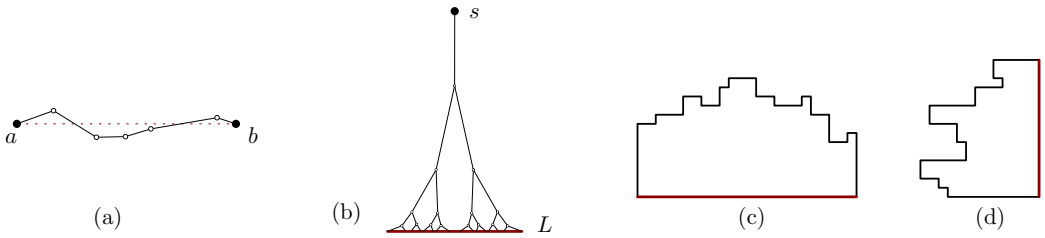


Figure 1 (a) A $(0\pm\delta)$-angle-bounded path. (b) A shallow-light tree between a source s and a horizontal line segment L. (c)–(d) An x- and a y-monotone histogram.

Shallow-Light Trees. Shallow-light trees (SLT) were introduced by Awerbuch et al. [5] and Khuller et al. [32]. Given a source s and a point set S in a metric space, an (α,β)-SLT is a Steiner tree rooted at s that contains a path of weight at most $\alpha\,\|ab\|$ between the *source* s and any point $t \in S$, and has weight at most $\beta\,\|\mathrm{MST}(S)\|$. We build on the following basic variant of SLT between a source s and a set S of collinear points in the plane; see Fig. 1(b).

▶ **Lemma 4** (Solomon [47, Section 2.1])**.** *Let $0<\varepsilon<1$, let $s = (0,\varepsilon^{-1/2})$ be a point on the y-axis, and let S be a set of points in the line segment $L = [-\frac{1}{2},\frac{1}{2}]\times\{0\}$ in the x-axis. Then there exists a geometric graph of weight $O(\varepsilon^{-1/2})$ that contains, for every point $t \in L$, an st-path P_{st} with $\|P_{st}\| \le (1+\varepsilon)\,\|st\|$.*

We note that the weight analysis of the st-path P_{st} in an SLT does not use angle-boundedness. In particular, an SLT may contain short edges of arbitrary directions close to t, but the directions of long edges are close to vertical. In Section 3 below, we generalize the shallow-light trees to obtain $(1+\varepsilon)$-spanners between points on two staircase paths.

Stretch Factor of $1+\varepsilon$ Versus $1+O(\varepsilon)$. In the geometric spanners we construct, an st-path may comprise $O(1)$ subpaths, each of which is angle-bounded or contained in an SLT. For the ease of presentation, we typically establish a stretch factor of $1+O(\varepsilon)$ in our proofs. It is understood that $1+\varepsilon$ can be achieved by a suitable scaling of the constant coefficients.

Histogram Decomposition. A path in the plane is *x-monotone* (resp., *y-monotone*) if its intersection with every vertical (resp., horizontal) line is connected. A *histogram* is a rectilinear simple polygon bounded by an axis-parallel line segment and an x- or y-monotone path; see Fig. 1(c–d). It is well known that every rectilinear simple polygon P can be subdivided into histograms (faces) such that every axis-parallel line segment in P intersects (*stabs*) at most three histograms [21, 36]; such a subdivision is also called a *window partition* [37, 48] of P, and can be computed in $O(n \log n)$ time if P has n vertices. The stabbing property implies that the total perimeter of the histograms in such a subdivision is $O(\mathrm{per}(P))$.

Dumitrescu and Tóth [20] showed that for a finite point set $S \subset \mathbb{R}^2$, one can refine the window partition, while increasing the weight by a constant factor, to construct a graph with constant geometric dilation. The *geometric dilation* of a geometric graph G is $\sup_{a,b\in G} d_G(a,b)/\|ab\|$, where $d_G(a,b)$ denotes the Euclidean length of a shortest path in G, and the supremum is taken over all point pairs $\{a,b\}$ at vertices and along edges of G. We follow a similar approach here, but we construct a subdivision of "modified" histograms (defined in Section 6), where the vertical edges are replaced by angle-bounded paths.

3 Generalized Shallow Light Trees

In Section 3.1, we generalize Lemma 4, and construct shallow-light trees between a source s and points on an x- and y-monotone rectilinear path L, which is called a *staircase path*. In Section 3.2, we show how to combine two shallow-light trees to obtain a spanner between point pairs on two staircase paths.

3.1 Single Source and Staircase Chain

We present a new, slightly modified proof for Solomon's result on SLTs between a single source s and a horizontal line segment, and then adapt the modified proof to obtain a SLT between s and an x- and y-monotone polygonal chain. In the proof below, we use the Taylor estimates $\cos x \geq 1 - x^2/2$ and $\sin x \geq x/2$ for $x \leq \pi/3$.

Alternative proof for Lemma 4. Assume w.l.o.g. that $\varepsilon = 2^{-k}$ for $k \in \mathbb{N}$. Let $T = \{t_i : i = 1, \ldots, 2^{k+1}\}$ be 2^{k+1} points on the line segment $L = [-\frac{1}{2}, \frac{1}{2}] \times \{0\}$ with uniform $1/(2^{k+1}-1) < \varepsilon$ spacing between consecutive points. Consider the standard binary partition of $\{1, \ldots, 2^{k+1}\}$ into intervals, associated with a binary tree: At level 0, the root corresponds to the interval $[1, 2^{k+1}]$ of all 2^{k+1} integer. At level j, we have intervals $[i\cdot 2^{k-j}+1, (i+1)\cdot 2^{k-j}]$ for $i = 0, \ldots, 2^j$. Note that if a point q is the left (resp., right) endpoint of an interval at a level j, then it is a left (resp., right) endpoint of all descendant intervals that contains it.

For every $q \in \{1, \ldots, 2^{k+1}\}$, we define a line segment ℓ_q with one endpoint at t_q: Let $j \geq 0$ be the smallest level such that q is an endpoint of some interval I_q at level j. If q is the left (resp., right) endpoint of I_q, then let ℓ_q be the line segment of direction $\frac{\pi}{2} - 2^{(j-k)/2}$

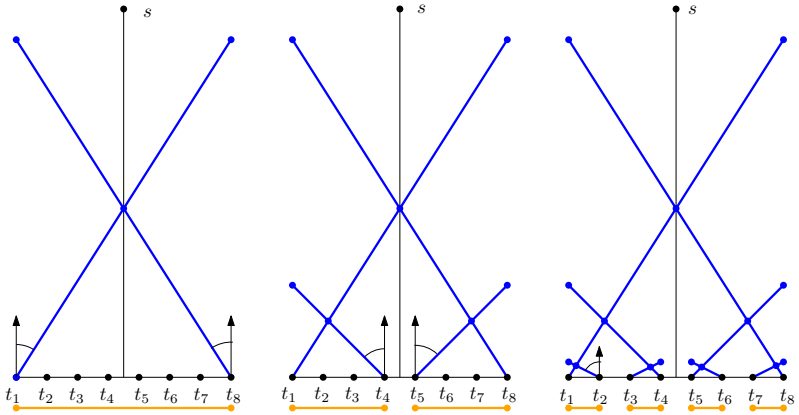


Figure 2 The segments added to graph G at level $j = 0, 1, 2$ for $m = 2^3 = 8$ points. The intervals $[t_a, t_b]$ at level j are indicated below the line L.

(resp., $\frac{\pi}{2} + 2^{(j-k)/2}$) such that its orthogonal projection to the x-axis is I_q; see Fig. 2. Note that for $j = 0$, we use directions $\frac{\pi}{2} \pm 2^{-k/2} = \frac{\pi}{2} \pm \sqrt{\varepsilon}$. Let G be the union of segments ℓ_q for $q = 1, \ldots, 2^{k+1}$, the horizontal segment L, and the vertical segment from s to the origin.

Lightness analysis. We show that $\|G\| = O(\varepsilon^{-1/2})$. We have $\|L\| = 1$, and the length of the vertical segment between s and the origin is $\varepsilon^{-1/2}$. At level j of the binary tree, we construct 2^j segments ℓ, each of length $\|\ell\| \le 2^{-j}/\sin(2^{(j-k)/2}) \le 2 \cdot 2^{(k-3j)/2}$. Summation over all levels yields $\sum_{j=0}^{k} 2^j \cdot 2 \cdot 2^{(k-3j)/2} = 2^{k/2} \cdot 2 \cdot \sum_{j=0}^{k} 2^{-j/2} = O(2^{k/2}) = O(\varepsilon^{-1/2})$.

Source-stretch analysis. We show that G contains an st_q-path of length $(1 + O(\varepsilon))\|st_q\|$ for all $q = 1, \ldots, 2^{k+1}$. First note that $\|st_q\| \ge \varepsilon^{-1/2}$, as the distance between s and L is $\varepsilon^{-1/2}$. For each interval $[t_a, t_b]$ in the binary tree, ℓ_a and ℓ_b have positive and negative slopes, respectively, and so they cross above the interval $[t_a, t_b]$. Consequently, for every point t_q, the union of the $k + 1$ segments corresponding to the intervals that contain t_q must contain a y-monotonically increasing path P_q from t_q to s. The y-projection of this path has length $\varepsilon^{-1/2}$. Consider one edge e of P_q along a segment ℓ at level j, which has direction $\frac{\pi}{2} \pm \alpha = \frac{\pi}{2} \pm 2^{(j-k)/2}$. Then the difference between the length of e and the y-projection of e is $\|e\|(1 - \cos\alpha) \le \|\ell\|(1 - \cos\alpha) \le 2^{-j} \frac{1-\cos\alpha}{\sin\alpha} \le 2^{-j} \frac{\alpha^2/2}{\alpha/2} = 2^{-j}\alpha = 2^{-j} \cdot 2^{(j-k)/2} = 2^{-(j+k)/2}$. Since P_q contains at most one edge in each level, summation over all edges of P_q yields $\sum_{j=0}^{k} 2^{-(j+k)/2} = 2^{-k/2} \sum_{j=0}^{k} 2^{-j/2} = O(\varepsilon^{1/2}) \le \|st_q\| \cdot O(\varepsilon)$.

Finally, for an arbitrary point $t \in L$, we have $\|st\| \ge \varepsilon^{-1/2}$, and G contains an st-path that consists of an st_q-path from s to the point t_q closest to t, followed by the horizontal segment $t_q t$ of weight $\|t_q t\| \le 1/2^k \le \varepsilon$. The total weight of this path is $(1 + O(\varepsilon))\|st\|$. After suitable scaling of the constant coefficients, G contains a path of weight at most $(1 + \varepsilon)\|st\|$ for any $t \in L$, as required. ◀

▶ **Lemma 5.** *Let $0 < \varepsilon < 1$, let $s = (0, \varepsilon^{-1/2})$ be a point on the y-axis, and let L be an x- and y-monotone increasing staircase path between the vertical lines $x = \pm\frac{1}{2}$, such that the right endpoint of L is $(\frac{1}{2}, 0)$ on the x-axis. Then there exists a geometric graph G comprised of L and additional edges of weight $O(\varepsilon^{-1/2})$ such that G contains, for every $t \in L$, an st-path P_{st} with $\|P_{st}\| \le (1 + O(\varepsilon))\,\|st\|$.*

We can adjust the construction above as follows; refer to Fig. 3.

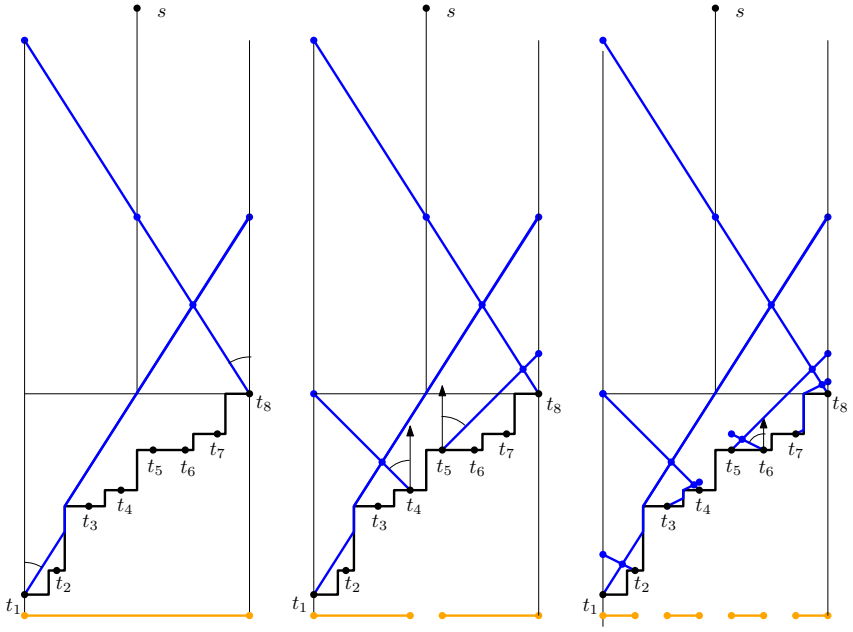


Figure 3 The paths γ_q added to graph G at level $j = 0, 1, 2$ for $m = 2^3 = 8$ points. The intervals $[t_a, t_b]$ at level j are indicated below the staircase path L.

Proof. Assume w.l.o.g. that $\varepsilon = 2^{-k}$ for some $k \in \mathbb{N}$. Let $T = \{t_i : i = 1, \ldots, 2^{k+1}\}$ be 2^{k+1} points in L on equally spaced vertical lines, with spacing $1/(2^{k+1} - 1) < \varepsilon$. Consider the standard binary partition of $\{1, \ldots, 2^{k+1}\}$ into intervals as in the previous proof.

For every $q \in \{1, \ldots, 2^{k+1}\}$, we define a polygonal path γ_q with one endpoint at t_q; see Fig. 3. Let $j \geq 0$ be the smallest level such that t_q is an endpoint of some interval I_q at level j. If t_q is the right endpoint of I_q, then let γ_q be the line segment of direction $\frac{\pi}{2} + 2^{(j-k)/2}$ such that its x-projection is I_q. If t_q is the left endpoint of I_q, then γ_q will be an x- and y-monotone path whose x-projection is I_q, and its edges will be vertical segments along L and segments of direction $\alpha_q = \frac{\pi}{2} - 2^{(j-k)/2}$. Specifically, γ_q starts from t_q with a line of direction α_q. Whenever γ_q encounters a vertical edge of L, it follows it upward until its upper endpoint, and then continues in direction α_q.

Let G be the union of all paths γ_q for $q = 1, \ldots, 2^{k+1}$, as well as the path L, and the vertical segment from s to the origin. This completes the construction of G.

Lightness analysis. We show that $\|G\| = \|L\| + O(\varepsilon^{-1/2})$. The distance between s and L is $\varepsilon^{-1/2}$. For every $q \in \{1, \ldots, 2^{k+1}\}$, the path γ_q is composed of vertical segments along L, and nonvertical segments whose total weight is the same as $\|\ell_q\|$ in the proof of Lemma 4, where we have seen that $\sum_{q=1}^{2^{k+1}} \|\ell_q\| = O(\varepsilon^{-1/2})$. Consequently, $\|G\| = \|L\| + O(\varepsilon^{-1/2})$.

Source-stretch analysis. We show that G contains an st_q-path of weight $(1 + (\varepsilon))\|st_q\|$ for all $q = 1, \ldots, 2^{k+1}$. Denoting $y(t_q)$ the y-coordinate of point t_q, we have $\|st_q\| \geq \varepsilon^{-1/2} + |y(t_q)|$. For each interval $[t_a, t_b]$ in the binary tree, the paths γ_a and γ_b cross above the portion of L between t_a and t_b. Consequently, for every point t_q, the union of the $k + 1$ paths γ corresponding to the intervals that contain t_q must contain a y-monotonically increasing path P_q from t_q to s. The y-projection of this path has length $\varepsilon^{-1/2} + |y(t_q)|$. Some of the edges of this path are vertical. Consider the union of all nonvertical edges e of P_q along a path γ at level j, which all have direction $\frac{\pi}{2} \pm 2^{(j-k)/2}$. The difference between the length of e and the y-projection of e is bounded by the same analysis as in the proof of Lemma 4. Summation over all levels yields $O(\varepsilon^{1/2}) \leq \|st_q\| \cdot O(\varepsilon)$.

Finally, for an arbitrary point $t \in L$, we have $\|st\| \geq |y(s) - y(t)| = \varepsilon^{-1/2} + |y(t)|$, and G contains an st-path that consists of an st_q-path from s to the closest point t_q to the right of t, followed by an x- and y-monotone path along L in which the total length of the horizontal edges is bounded by $1/2^k \leq \varepsilon$ (and the length of vertical segment might be arbitrary). We use the lower bound $\|st\| \geq \varepsilon^{-1/2} + |y(t)|$. The vertical segments between t_q and t do not contribute to the error term $\|st\| - (\varepsilon^{-1/2} + |y(t)|)$. The analysis in the proof of Lemma 4 yields $\|st\| - (\varepsilon^{-1/2} + |y(t)|) \leq O(\sqrt{\varepsilon}) \leq O(\varepsilon)\|st\|$. ◀

Note that the source-stretch analysis assumed that the vertical edges of an st-path (along the vertical edges of L) do not accumulate any error. Consequently, the same analysis carries over if we replace the vertical edges of L by $(\frac{\pi}{2} \pm \frac{\sqrt{\varepsilon}}{2})$-angle-bounded paths. The key observation is that in the proof of Lemma 5, all nonvertical edges have directions that differ from vertical (i.e., from $\frac{\pi}{2}$) by $\sqrt{\varepsilon}$ or more.

▶ **Corollary 6.** *Let $0 < \varepsilon < 1$, let $s = (0, \varepsilon^{-1/2})$ be a point on the y-axis, and let L be a path between the vertical lines $x = \pm\frac{1}{2}$, obtained from an x- and y-monotone increasing staircase path with the right endpoint at $(\frac{1}{2}, 0)$ on the x-axis, by replacing the vertical edges with y-monotonically increasing $(\frac{\pi}{2} \pm \frac{\sqrt{\varepsilon}}{2})$-angle-bounded paths. Then there exists a geometric graph G that contains L and additional edges of weight $O(\varepsilon^{-1/2})$ such that G contains, for every $t \in L$, an st-path P_{st} with $\|P_{st}\| \leq (1 + O(\varepsilon))\,\|st\|$.*

3.2 Combination of Shallow-Light Trees

The combination of two SLTs yields a light $(1+\varepsilon)$-spanner between points on two staircases.

▶ **Lemma 7.** *Let R be an axis-parallel rectangle of width 1 and height $2\varepsilon^{-1/2}$; and let L_1 (resp., L_2) be a staircase path from the lower-left (upper-left) corner of R to a point on the vertical line passing through the right side of R, lying below (above) R; see Fig. 4. Then there exists a geometric graph comprised of $L_1 \cup L_2$ and additional edges of weight $O(\varepsilon^{-1/2})$ that contains an ab-path P_{ab} with $\|P_{ab}\| \leq (1 + O(\varepsilon))\,\|ab\|$ for any $a \in L_1$ and any $b \in L_2$.*

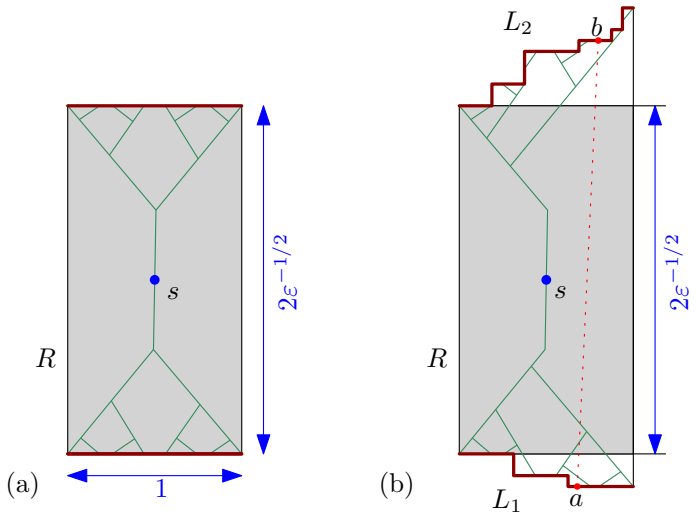


Figure 4 (a) A combination of two SLTs between the two horizontal sides of R. (b) A combination of two SLTs between two staircases above and below R, respectively.

Proof. Let s be the center of the rectangle R. Let G be the geometric graph formed by the SLTs from the source s to L_1 and L_2, resp., using Lemma 5. By construction, $\|G\| = \|L_1\| + \|L_2\| + O(\varepsilon^{-1/2})$. It remains to show that G has the desired spanning ratio. Let $a \in L_1$ and $b \in L_2$. Let h_a be the distance of a from bottom side of R, and h_b the distance of b from the top side of R. By Lemma 4, the two SLTs jointly contain an ab-path P_{ab} of length $\|P_{ab}\| \leq (1 + O(\varepsilon))\,(\|as\| + \|bs\|)$.

On the one hand, s is the center of R, and so $\|as\| + \|bs\| \leq \mathrm{diam}(R) + h_a + h_b \leq (1 + \frac{\varepsilon}{8})2\varepsilon^{-1/2} + h_a + h_b$. On the other hand, $\|ab\| \geq \mathrm{height}(R) + h_a + h_b = 2\varepsilon^{-1/2} + h_a + h_b$. Overall, $\|P_{ab}\| \leq (1 + O(\varepsilon))(1 + \frac{\varepsilon}{8})\,\|ab\| \leq (1 + O(\varepsilon))\|ab\|$. ◀

4 Reduction to Directional Spanners in Histograms

In this section, we present our general strategy for the proof of Theorem 2, and reduce the construction of a light $(1 + \varepsilon)$-spanner for a point set S in the plane to a special case of *directional* spanners for a point set on the boundary of faces in a (modified) *window partition*.

Directional $(1 + \varepsilon)$-Spanners. Our strategy to construct a $(1 + \varepsilon)$-spanner for a point set S is to partition the interval of directions $[0, \pi)$ into $O(\varepsilon^{-1/2})$ intervals, each of length $O(\varepsilon^{1/2})$. For each interval $D \subset [0, \pi)$, we construct a geometric graph that serves point pairs $a, b \in S$ with $\mathrm{dir}(ab) \in D$. Then the union of these graphs over all $O(\varepsilon^{-1/2})$ intervals will serve all point pairs $ab \in S$. The following definition formalizes this idea.

▶ **Definition 8.** *Let S be a finite point set in $\mathbb{R}^2$, and let $D \subset [0, \pi)$ be a set of directions. A geometric graph G is a* directional $(1 + \varepsilon)$-spanner *for S and D if G contains an ab-path of weight at most $(1 + \varepsilon)\|ab\|$ for every $a, b \in S$ with $\mathrm{dir}(ab) \in D$.*

In Section 6, we modify the standard window partition algorithm and partition a bounding box of S into *fuzzy staircases* and *tame histograms* (defined below). We also construct directional spanners for point pairs $a, b \in S$, where ab is a chord of a face in this partition. A line segment ab is a *chord* of a simple polygon P if $a, b \in \partial P$, and $ab \subset P$. The *perimeter* of a simple polygon P, denoted $\mathrm{per}(P)$, is the total weight of the edges of P; and the *horizontal perimeter*, denoted $\mathrm{hper}(P)$, is the total weight of the horizontal edges of P.

▶ **Lemma 9.** *We can subdivide a simple rectilinear polygon P into a collection $\mathcal{F}$ of fuzzy staircases and tame histograms of total perimeter $\sum_{F \in \mathcal{F}} \mathrm{per}(F) \leq O(\varepsilon^{-1/2}\mathrm{per}(P))$ and total horizontal perimeter $\sum_{F \in \mathcal{F}} \mathrm{hper}(F) \leq O(\mathrm{per}(P))$.*

▶ **Lemma 10.** *Let F be a fuzzy staircase or a tame histogram, $S \subset \partial F$ a finite point set, $\varepsilon > 0$, and $D = [\frac{\pi - \sqrt{\varepsilon}}{2}, \frac{\pi + \sqrt{\varepsilon}}{2}]$ an interval of nearly vertical directions. Then there exists a geometric graph of weight $O(\mathrm{per}(F) + \varepsilon^{-1/2}\,\mathrm{hper}(F))$ such that for all $a, b \in S$, if ab is a chord of F and $\mathrm{dir}(ab) \in D$, then G contains an ab-path of weight at most $(1 + O(\varepsilon))\|ab\|$.*

For the proof of Lemmas 9 and 10, refer to the full paper [6]. In the remainder of this section, we show that these lemmas imply Lemma 11, which in turn implies Theorem 2.

▶ **Lemma 11.** *Let $S \subset \mathbb{R}^2$ be a finite point set, $\varepsilon > 0$, and $D \subset [0, \pi)$ an interval of length $\sqrt{\varepsilon}$. Then there exists a* directional $(1 + \varepsilon)$-spanner *for S and D of weight $O(\varepsilon^{-1/2}\,\|MST\|)$.*

Proof. We may assume, by applying a suitable rotation, that $D = [\frac{\pi - \sqrt{\varepsilon}}{2}, \frac{\pi + \sqrt{\varepsilon}}{2}]$, that is, an interval of nearly vertical directions. We construct a directional $(1 + \varepsilon)$-spanner for S and D of weight $O(\varepsilon^{-1/2} \cdot \|\mathrm{MST}(S)\|)$.

Assume w.l.o.g. that the unit square $U = [0,1]^2$ is the minimum axis-parallel bounding square of S. In particular, S has two points on two opposite sides of U, and so $1 \leq \mathrm{diam}(S) \leq \|\mathrm{MST}(S)\|$. Our initial graph G_0 is composed of the boundary of U and a *rectilinear MST* of S, where $\|G_0\| = O(\|\mathrm{MST}(S)\|)$. Since each edge of G_0 is on the boundary of at most two faces, the total perimeter of all faces of G_0 is also $O(\|\mathrm{MST}(S)\|)$. Lemma 9 yields subdivisions of the faces of G_0 into a collection $\mathcal{F}$ of fuzzy staircases and tame histograms of total perimeter $\sum_{F\in\mathcal{F}} \mathrm{per}(F) = O(\varepsilon^{-1/2}\|MST(S)\|)$ and horizontal perimeter $\sum_{F\in\mathcal{F}} \mathrm{hper}(F) = O(\|MST(S)\|)$,

Let $K(S)$ be the complete graph induced by S. For each face $F \in \mathcal{F}$, let S_F be the set of all intersection points between the boundary ∂F and the edges of $K(S)$. For each face F, Lemma 10 yields a geometric graph G_F of weight $O(\mathrm{per}(F) + \varepsilon^{-1/2}\mathrm{hper}(F))$ with respect to the finite point set $S_F \subset \partial F$.

We can now put the pieces back together. Let G be the union of G_0 and the graphs G_F for all $F \in \mathcal{F}$. The weight of G is bounded by $\|G\| = \|G_0\| + \sum_{F\in\mathcal{F}} \|G_F\| = O(\|\mathrm{MST}(S)\| + \sum_{F\in\mathcal{F}}(\mathrm{per}(F) + \varepsilon^{-1/2}\mathrm{hper}(F))) = O(\varepsilon^{-1/2}\|\mathrm{MST}(S)\|)$.

Let $a, b \in S$. The edges of G_0 subdivide the line segment ab into a path $v_0v_1 \ldots v_m$ of collinear segments, each of which is a chord of some face in $\mathcal{F}$. For $i = 1, \ldots, m$, graph G contains a $v_{i-1}v_i$-path of weight at most $(1+\varepsilon)\|v_{i-1}v_i\|$. The concatenation of these paths is an ab-path of length at most $\sum_{i=1}^{m}(1+\varepsilon)\|v_{i-1}v_i\| = (1+\varepsilon)\|ab\|$, as required. ◀

Proof of Theorem 2. Let S be a finite set of points in the plane. Let $\varepsilon > 0$ be given. For $k = \lceil \pi\varepsilon^{-1/2} \rceil$, we partition the space of directions as $[0, \pi) = \bigcup_{i=1}^{k} D_i$, into k intervals of equal length. By Lemma 11, there exists a directional $(1+\varepsilon)$-spanner of weight $O(\varepsilon^{-1/2}\|\mathrm{MST}(S)\|)$ for S and D_i for all i. Let $G = \bigcup_{i=1}^{k} G_i$. For every point pair $s, t \in S$, we have $\mathrm{dir}(st) \in D_i$ for some $i \in \{1, \ldots, k\}$, and $G_i \subset G$ contains an st-path of weight at most $(1+\varepsilon)\|st\|$. Consequently, G is a Euclidean Steiner $(1+\varepsilon)$-spanner for S. The weight of G is $\|G\| = \sum_{i=1}^{k}\|G_i\| \leq \lceil \pi\varepsilon^{-1/2} \rceil \cdot O(\varepsilon^{-1/2}\|\mathrm{MST}(S)\|) \leq O(\varepsilon^{-1}\|\mathrm{MST}(S)\|)$, as required. ◀

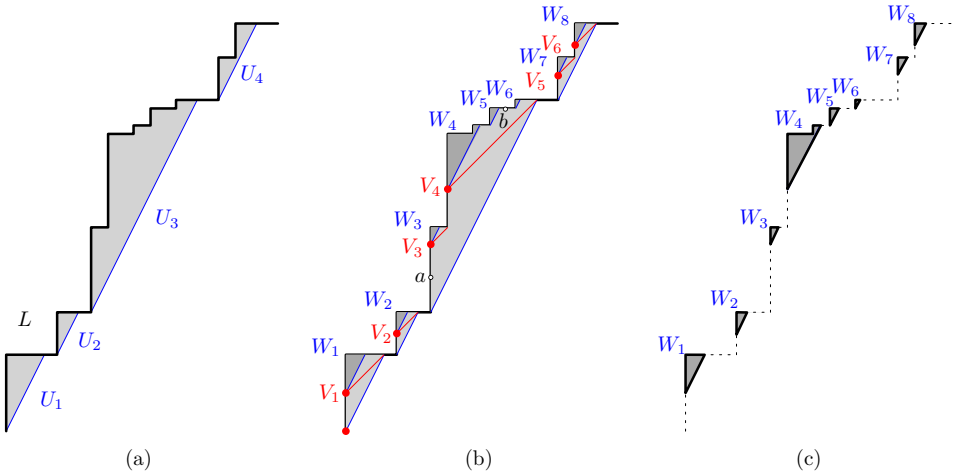
5 Construction of Directional Spanners for Staircases

In this section, we handle the special case where the points are on a x- and y-monotone rectilinear path L, which is called a *staircase path*. Our recursive construction uses a type of polygons that we define now. A *shadow polygon* is bounded by a staircase path L and a single line segment of slope $\varepsilon^{-1/2}$; see Fig. 5(a) for examples.

▶ **Lemma 12.** *Let L be an x- and y-monotonically increasing staircase path, and let $S \subset L$ be a finite point set. Then there exists a geometric graph G comprised of L and additional edges of weight $O(\varepsilon^{-1/2}\mathrm{width}(L))$ such that G contains a path P_{ab} of weight $\|P_{ab}\| \leq (1+O(\varepsilon))\|ab\|$ for any $a, b \in L$ where $\mathrm{slope}(ab) \geq \varepsilon^{-1/2}$ and the line segment ab lies below L.*

Proof. If $a, b \in L$ and ab lies below L, then either both a and b are in the same edge of L (hence L contains a straight-line path ab), or one point in $\{a, b\}$ is on a vertical edge of L and the other is on a horizontal edge of L. We may assume w.l.o.g. that a is on a vertical edge and b is on a horizontal edge of L.

Let A be the set of all points p such that there exists $a \in L$ on some vertical edge of L such that $\mathrm{slope}(ap) \geq \varepsilon^{-1/2}$ and ap is below L; see Fig. 5(a). The set A is not necessarily connected, the connected components of A are shadow polygons for disjoint subpaths of L. Let $\mathcal{U}$ be the set of these shadow polygons. Note that for every pair $a, b \in L$, if $\mathrm{slope}(ab) \geq \varepsilon^{-1/2}$ and ab lies below the path L, then ab lies in some polygon in $\mathcal{U}$. For each polygon $U \in \mathcal{U}$, we construct a geometric graph $G(U)$ of weight $O(\varepsilon^{-1/2}\mathrm{width}(U))$ such



(a) (b) (c)

Figure 5 (a) A staircase path L; the shadow of vertical edges of L is shaded light gray. (b) The shadow of the horizontal edges in the subpolygons is shaded dark gray. (c) Recursive subproblems generated in the proof of Lemma 12.

that $G(U) \cup L$ is a directional spanner for the point pairs in $S \cap U$. Then L together with $\bigcup_{U\in\mathcal{U}} G(U)$ is a directional spanner for all possible ab pairs. Since the shadow polygons in $\mathcal{U}$ are adjacent to disjoint portions of L, we have $\sum_{U\in\mathcal{U}} \text{width}(U) \leq \text{width}(L)$, and so $\sum_{U\in\mathcal{U}} \|G(U)\| = O(\varepsilon^{-1/2}\text{width}(L))$, as required.

Recursive Construction. For all $U \in \mathcal{U}$, we construct $G(U)$ recursively as follows. Assume that $|S \cap U| \geq 2$. Let $B(U)$ be the set of all points $p \in U$ for which there exists a point b on some horizontal edge of U such that $bp \subset U$ and $\text{slope}(ab) \geq \frac{1}{2}\varepsilon^{-1/2}$; see Fig. 5(b). The set $B(U)$ may be disconnected, each component is a simple polygon bounded by a contiguous portion of L and a line segment of slope $\frac{1}{2}\varepsilon^{-1/2}$. Denote by $\mathcal{V}$ the set of connected components of $B(U)$.

For every $V \in \mathcal{V}$, let $C(V)$ be the set of all points $p \in V$ for which there exists a point a on some vertical edge of V such that $ap \subset V$ and $\text{slope}(ap) \geq \varepsilon^{-1/2}$; see Fig. 5(b). Again, the set $C(V)$ may be disconnected, each component is a shadow polygon. Denote by $\mathcal{W}$ the set of all connected components of $C(V)$ for all $V \in \mathcal{V}$.

Since $\text{height}(W)/\text{width}(W) = \varepsilon^{-1/2}$ for all $W \in \mathcal{W}$ and $\text{height}(V)/\text{width}(V) = \frac{1}{2}\varepsilon^{-1/2}$ for all $V \in \mathcal{V}$, we have

$$\sum_{W\in\mathcal{W}} \text{width}(W) = \sqrt{\varepsilon} \cdot \sum_{W\in\mathcal{W}} \text{height}(W) = \sqrt{\varepsilon} \cdot \sum_{V\in\mathcal{V}} \text{height}(V)$$
$$= \frac{1}{2} \sum_{V\in\mathcal{V}} \text{width}(V) = \frac{1}{2} \sum_{U\in\mathcal{U}} \text{width}(U). \tag{1}$$

For every polygon $V \in \mathcal{V}$, let s_V be the bottom vertex of V. We construct a sequence of shallow-light trees from source s_V as follows. For every nonnegative integer $i \geq 0$, let h_i be a horizontal line at distance $\text{height}(V)/2^i$ above s_V. If there is any point in S between h_i and h_{i+1}, then we construct an SLT from s_V to the portion of L between h_i and h_{i+1}. By Lemma 12, the total weight of these trees is $O(\varepsilon^{-1/2}\text{width}(V))$. Over all $V \in \mathcal{V}$, the weight of these SLTs is $\sum_{V\in\mathcal{V}} O(\varepsilon^{-1/2}\text{width}(V)) = O(\varepsilon^{-1/2}\text{width}(U))$. For all $V \in \mathcal{V}$, we also add the boundary ∂V to our spanner, at a cost of $\sum_{V\in\mathcal{V}} \text{per}(V) = \sum_{V\in\mathcal{V}} O(\varepsilon^{-1/2}\text{width}(V)) = O(\varepsilon^{-1/2}\text{width}(U))$. This completes the description of one iteration. Recurse on all $W \in \mathcal{W}$ that contain any point in S.

Lightness analysis. Each iteration of the algorithm, for a shadow polygon U, constructs SLTs of total weight $O(\varepsilon^{-1/2}\text{width}(U))$, and produces subproblems whose combined width is at most $\frac{1}{2}\text{width}(U)$ by Equation (1). Consequently, summation over all levels of the recursion yields $\|G(U)\| = O(\varepsilon^{-1/2}\text{width}(U) \cdot \sum_{i\geq 0} 2^{-i}) = O(\varepsilon^{-1/2}\text{width}(U))$, as required.

Stretch analysis. Now consider a point pair $a, b \in S$ such that $\text{slope}(ab) \geq \varepsilon^{-1/2}$, a is in a vertical edge of L, and b is in a horizontal edge of L. Assume that U is the smallest shadow polygon in the recursive algorithm above that contains both a and b. Then $b \in V$ for some $V \in \mathcal{V}$, and a is at or below vertex s_V of V. Now we can find an ab-path P_{ab} as follows: First construct a y-monotonically increasing path from a to V_S along vertical edges of L and along edges of some polygons in $\mathcal{V}$; all these edges have slope larger than $\frac{1}{2}\varepsilon^{-1/2}$. Then from s_V, follow an SLT to b. All edges of P_{ab} from a to s_V have slope at least $\frac{1}{2}\varepsilon^{-1/2}$, and so their directions differ from vertical by at most $\arctan(2\varepsilon^{1/2}) \leq 3\varepsilon^{1/2}$, using the Taylor expansion of $\tan(x)$ near 0. By Lemma 3 the stretch factor of the paths from a to s_V and the path as_Vb are each at most $1 + O(\varepsilon)$. By Lemma 12, the SLT contains a path from s_V to b with stretch factor $1 + O(\varepsilon)$. Overall, $\|P_{ab}\| \leq (1 + O(\varepsilon))\|ab\|$. ◀

In the full paper [6], it is shown that Lemma 12 continues to hold if we replace the vertical edges of the staircase L with angle-bounded paths. Furthermore, the horizontal edges can also be replaced by x-monotone paths of approximately the same length.

6 Construction of Directional Spanners in Histograms

We would like to partition a simple rectilinear polygon P into a collection $\mathcal{F}$ of simple polygons (faces), and then design a directional $(1+\varepsilon)$-spanner for each face $F \in \mathcal{F}$ such that the total weight of these spanners is under control. Lemma 12 tells us that we can handle staircase polygons efficiently. The standard window partition [37, 48] would partition P into histograms as indicated in Fig. 6(a). We would like to further reduce the problem to staircase polygons. However, the worst-case weight of a standard decomposition of a histogram H into staircases is $\Theta(\text{per}(H)\log n)$, where n is the number of vertices of H. We cannot afford a $\log n$ factor (or any function of the cardinality $|S|$). To overcome this technical difficulty, we replace the vertical edges by nearly vertical δ-angle-bounded paths (cf. Lemma 3). By setting $\delta = \Theta(\sqrt{\varepsilon})$, these paths provide enough flexibility to keep the weight of the subdivision under control; and our result on SLTs for these "modified" staircases carry over with only a constant increase in their total weight.

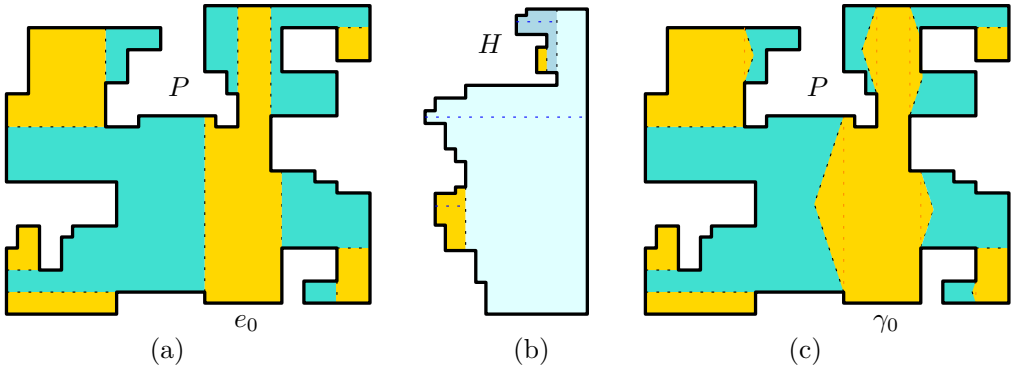


Figure 6 (a) A standard window partition of a rectilinear polygon P into histograms, starting from a horizontal edge e_0. (b) A decomposition of a y-monotone histogram into staircase polygons. (c) The modified window partition of a rectilinear polygon P into x-monotone Λ-histograms and y-monotone fuzzy histograms.

We introduce some terminology; refer to Fig. 7. Let $\Lambda \geq 8$ be a constant.

- A Λ-*path* is a y-monotone path in which every edge is vertical, or has slope $\pm\Lambda\varepsilon^{-1/2}$.
- A Λ-*histogram* is a simple polygon obtained from a histogram by replacing vertical edges with some Λ-paths. A Λ-histogram is *x-monotone* (resp., *y-monotone*) if it is obtained from an x-monotone (resp., y-monotone) histogram.
- A *fuzzy staircase* is a simple polygon bounded by a path pqr, where pq is horizontal and $\text{slope}(qr) = \pm\Lambda\varepsilon^{-1/2}$, and a pr-path obtained from an x- and y-monotone staircase by replacing vertical edges with some Λ-paths.
- A *fuzzy histogram* is a simple polygon bounded by a y-monotone rectilinear path L and a path γ of one or two edges of slopes $\pm\Lambda\varepsilon^{-1/2}$; if the latter path has two edges, then its interior vertex is a reflex vertex of the polygon.
- A *tame histogram* (Fig. 8(a)) is a simple polygon bounded by a horizontal line segment pq and an pq-path L that consists of ascending or descending Λ-paths and x-monotone increasing horizontal edges with the following properties: (i) there is no chord between interior points of any two ascending (resp., descending) Λ-paths; (ii) for every horizontal chord ab, with $a, b \in L$, the subpath L_{ab} of L between a and b satisfies $\|L_{ab}\| \leq 2\|ab\|$.
- A *tame path* is a subpath of the pq-path L of a tame histogram.

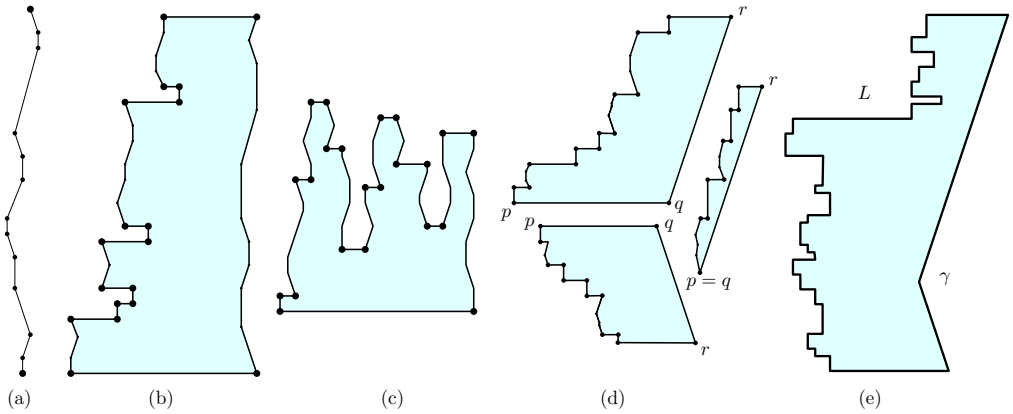


Figure 7 (a) A Λ-path. (b) A y-monotone Λ-histogram. (c) An x-monotone Λ-histogram. (d) Three fuzzy staircases. (e) A fuzzy histogram.

In what follows, we describe our spanner constructions in five modules. However, due to space constraints we provide only an overview of these modules and refer to the full version of the paper [6] for the formal description and the proofs.

Fuzzy Window Decomposition. Let R be a rectilinear simple polygon. By modifying the standard window-partition, we show how to partition R into a collection $\mathcal{H}$ of x-monotone Λ-histograms and y-monotone fuzzy histograms such that $\sum_{H\in\mathcal{H}} \text{per}(H) = O(\text{per}(P))$; see Fig. 6(b). Furthermore, we show that in the x-monotone Λ-histograms, there is no chord between interior points of two ascending (resp., two descending) Λ-paths.

***y*-Monotone Histograms.** Let H be a (y-monotone) fuzzy histogram. We recursively subdivide H into a family $\mathcal{F}$ of fuzzy staircases using subdivision edges of total weight $O(\varepsilon^{-1/2}\,\text{per}(H))$ such that $\sum_{F\in\mathcal{F}} \text{hper}(F) = O(\text{per}(H))$; see Fig. 8(b) for an illustration.

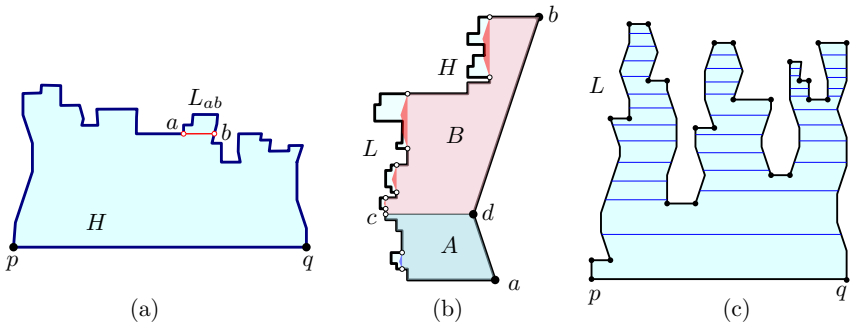


Figure 8 (a) A tame histogram. (b) Recursive subdivision of a fuzzy histogram into fuzzy staircase polygons. (c) Recursive subdivision of an x-monotone Λ-histograms into tame histograms.

x-Monotone Λ-Histograms. Let H be an x-monotone Λ-histogram that does not have any chords between interior points of any two ascending (resp., two descending) Λ-paths. We use a sweepline algorithm to subdivide H into tame histograms; see Fig. 8(c) for an illustration. Specifically, we subdivide H into a collection $\mathcal{T}$ of tame histograms such that $\sum_{T\in\mathcal{T}} \mathrm{per}(T) = O(\mathrm{per}(H))$. This module provides a proof for Lemma 9.

Directional Spanners for Tame Histograms. Given a tame histogram H and a finite set of points $S \subset \partial H$, we construct a directional spanner for S with respect to point pairs $a, b \in S$ with $|\mathrm{slope}(ab)| \geq \varepsilon^{-1/2}$. First we adapt Lemma 12 to construct a directional spanner for points $a, b \in S$ on a tame path $L \subset \partial H$; and then generalize Lemma 7 to handle point pairs where a is in the horizontal base of H and $b \in L$.

Directional Spanners for Fuzzy Staircases. Given a fuzzy staircase polygon F and a finite point set $S \subset \partial F$, we construct a directional spanner of weight $\|G\| = O(\varepsilon^{-1/2}\,\mathrm{hper}(F))$ for S with respect to point pairs $a, b \in S$ with $|\mathrm{slope}(ab)| \geq \varepsilon^{-1/2}$. The last two modules jointly imply Lemma 10, and complete all components needed for Theorem 2.

7 Conclusion and Outlook

We have proved a tight upper bound of $O(\varepsilon^{-1})$ on the lightness of Euclidean Steiner $(1+\varepsilon)$-spanners in the plane. That is, for every finite set $S \subset \mathbb{R}^2$, there is a Euclidean Steiner $(1+\varepsilon)$-spanner of weight $O(\varepsilon^{-1}\,\|\mathrm{MST}(S)\|)$. Our proof is constructive, but we do not control the number of Steiner points. This immediately raises the question about the optimum number of Steiner points: What is the minimum sparsity of a Euclidean Steiner $(1+\varepsilon)$-spanner of weight $O(\frac{1}{\varepsilon}\,\|\mathrm{MST}(S)\|)$ that can be attained for all finite set of points in $\mathbb{R}^2$?

Planarity is an important aspect of any geometric networks. Therefore, it is desirable to construct Euclidean $(1+\varepsilon)$-spanners that are planar, i.e., no two edges of the spanner cross. Any Steiner spanner can be turned into a plane spanner (planarized), with the same weight and the same spanning ratio between the input points, by introducing Steiner points at all edge crossings. However, planarization may substantially increase the number of Steiner points. Bose and Smid [10, Sec. 4] note that Arikati et al. [2] constructed a Euclidean plane $(1+\varepsilon)$-spanner with $O(\varepsilon^{-4}n)$ Steiner points for any n points in $\mathbb{R}^2$; see also [38]. Borradaile and Eppstein [8] improved the bound to $O(\varepsilon^{-3}n\log\varepsilon^{-1})$ in certain special cases where all Delaunay faces are fat. It remains an open problem to find the optimum dependence of ε for plane Steiner $(1+\varepsilon)$-spanners; and for plane Steiner $(1+\varepsilon)$-spanners of lightness $O(\varepsilon^{-1})$.

References

1 Ingo Althöfer, Gautam Das, David Dobkin, Deborah Joseph, and José Soares. On sparse spanners of weighted graphs. *Discrete & Computational Geometry*, 9(1):81–100, 1993.

2 Srinivasa Rao Arikati, Danny Z. Chen, L. Paul Chew, Gautam Das, Michiel H. M. Smid, and Christos D. Zaroliagis. Planar spanners and approximate shortest path queries among obstacles in the plane. In *Proc. 4th European Symposium on Algorithms (ESA)*, volume 1136 of *LNCS*, pages 514–528. Springer, 1996.

3 Sunil Arya, David M Mount, and Michiel Smid. Randomized and deterministic algorithms for geometric spanners of small diameter. In *Proc. 35th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 703–712, 1994.

4 Sunil Arya and Michiel Smid. Efficient construction of a bounded-degree spanner with low weight. *Algorithmica*, 17(1):33–54, 1997.

5 Baruch Awerbuch, Alan E. Baratz, and David Peleg. Cost-sensitive analysis of communication protocols. In *Proc. 9th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 177–187, 1990.

6 Sujoy Bhore and Csaba D. Tóth. Light Euclidean Steiner spanners in the plane. *Preprint*, 2020. `arXiv:2012.02216`.

7 Sujoy Bhore and Csaba D. Tóth. On Euclidean Steiner $(1+\varepsilon)$-spanners. In *Proc. 38th Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 187 of *LIPIcs*, pages 13:1–13:16. Schloss Dagstuhl, 2021.

8 Glencora Borradaile and David Eppstein. Near-linear-time deterministic plane Steiner spanners for well-spaced point sets. *Comput. Geom.*, 49:8–16, 2015.

9 Glencora Borradaile, Hung Le, and Christian Wulff-Nilsen. Greedy spanners are optimal in doubling metrics. In *Proc. 13th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2371–2379, 2019.

10 Prosenjit Bose and Michiel H. M. Smid. On plane geometric spanners: A survey and open problems. *Comput. Geom.*, 46(7):818–830, 2013.

11 Paul B. Callahan. Optimal parallel all-nearest-neighbors using the well-separated pair decomposition. In *Proc. 34th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 332–340, 1993.

12 Timothy M. Chan, Sariel Har-Peled, and Mitchell Jones. On locality-sensitive orderings and their applications. *SIAM J. Comput.*, 49(3):583–600, 2020.

13 Shiri Chechik and Christian Wulff-Nilsen. Near-optimal light spanners. *ACM Transactions on Algorithms (TALG)*, 14(3):1–15, 2018.

14 L. Paul Chew. There is a planar graph almost as good as the complete graph. In *Proc. 2nd Symposium on Computational Geometry*, pages 169–177. ACM Press, 1986.

15 L. Paul Chew. There are planar graphs almost as good as the complete graph. *J. Comput. Syst. Sci.*, 39(2):205–219, 1989.

16 Kenneth L. Clarkson. Approximation algorithms for shortest path motion planning. In *Proc. 19th ACM Symposium on Theory of Computing (STOC)*, pages 56–65, 1987.

17 Gautam Das, Paul Heffernan, and Giri Narasimhan. Optimally sparse spanners in 3-dimensional Euclidean space. In *Proc. 9th Symposium on Computational Geometry (SoCG)*, pages 53–62. ACM Press, 1993.

18 Gautam Das, Giri Narasimhan, and Jeffrey S. Salowe. A new way to weigh malnourished Euclidean graphs. In *Proc. 6th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 215–222, 1995.

19 Michael J. Demmer and Maurice P. Herlihy. The arrow distributed directory protocol. In *Proc. 12th Symposium on Distributed Computing (DISC)*, volume 1499 of *LNCS*, pages 119–133. Springer, 1998.

20 Adrian Dumitrescu and Csaba D. Tóth. Light orthogonal networks with constant geometric dilation. *J. Discrete Algorithms*, 7(1):112–129, 2009.

21 Herbert Edelsbrunner, Joseph O'Rourke, and Emmerich Welzl. Stationing guards in rectilinear art galleries. *Computer Vision, Graphics, and Image Processing*, 27(2):167–176, 1984.

22 Michael Elkin, Ofer Neiman, and Shay Solomon. Light spanners. *SIAM Journal on Discrete Mathematics*, 29(3):1312–1321, 2015.

23 Michael Elkin and Shay Solomon. Steiner shallow-light trees are exponentially lighter than spanning ones. *SIAM Journal on Computing*, 44(4):996–1025, 2015.

24 Jie Gao, Leonidas J. Guibas, and An Nguyen. Deformable spanners and applications. *Comput. Geom.*, 35(1-2):2–19, 2006.

25 Lee-Ad Gottlieb. A light metric spanner. In *2015 IEEE 56th Symposium on Foundations of Computer Science*, pages 759–772, 2015.

26 Lee-Ad Gottlieb, Aryeh Kontorovich, and Robert Krauthgamer. Efficient regression in metric spaces via approximate Lipschitz extension. *IEEE Transactions on Information Theory*, 63(8):4838–4849, 2017.

27 Joachim Gudmundsson, Christos Levcopoulos, and Giri Narasimhan. Fast greedy algorithms for constructing sparse geometric spanners. *SIAM J. Comput.*, 31(5):1479–1500, 2002.

28 Joachim Gudmundsson, Christos Levcopoulos, Giri Narasimhan, and Michiel Smid. Approximate distance oracles for geometric spanners. *ACM Transactions on Algorithms (TALG)*, 4(1):1–34, 2008.

29 Maurice Herlihy, Srikanta Tirthapura, and Rogert Wattenhofer. Competitive concurrent distributed queuing. In *Proc. 20th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 127–133, 2001.

30 J. Mark Keil. Approximating the complete Euclidean graph. In *Proc. 1st Scandinavian Workshop on Algorithm Theory (SWAT)*, volume 318 of *LNCS*, pages 208–213. Springer, 1988.

31 J. Mark Keil and Carl A. Gutwin. Classes of graphs which approximate the complete Euclidean graph. *Discrete & Computational Geometry*, 7:13–28, 1992.

32 Samir Khuller, Balaji Raghavachari, and Neal E. Young. Balancing minimum spanning and shortest path trees. In *Proc. 4th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 243–250, 1993.

33 Hung Le and Shay Solomon. Truly optimal Euclidean spanners. In *Proc. 60th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 1078–1100, 2019.

34 Hung Le and Shay Solomon. Light Euclidean spanners with Steiner points. In *Proc. 28th European Symposium on Algorithms (ESA)*, volume 173 of *LIPIcs*, pages 67:1–67:22. Schloss Dagstuhl, 2020.

35 Hung Le and Shay Solomon. A unified and fine-grained approach for light spanners. *CoRR*, abs/2008.10582, 2020. `arXiv:2008.10582`.

36 Christos Levcopoulos. *Heuristics for Minimum Decompositions of Polygons*. PhD thesis, Linköping, 1987. No. 74 of Linköping Studies in Science and Technology.

37 Anil Maheshwari, Jörg-Rüdiger Sack, and Hristo N. Djidjev. Link distance problems. In Jörg-Rüdiger Sacks and Jorge Urutia, editors, *Handbook of Computational Geometry*, chapter 12, pages 519–558. North-Holland, 2000.

38 Anil Maheshwari, Michiel H. M. Smid, and Norbert Zeh. I/O-efficient algorithms for computing planar geometric spanners. *Comput. Geom.*, 40(3):252–271, 2008.

39 Giri Narasimhan and Michiel Smid. *Geometric Spanner Networks*. Cambridge University Press, 2007.

40 David Peleg and Alejandro A. Schäffer. Graph spanners. *Journal of Graph Theory*, 13(1):99–116, 1989.

41 David Peleg and Jeffrey D. Ullman. An optimal synchronizer for the hypercube. *SIAM J. Comput.*, 18(4):740–747, 1989.

42 David Peleg and Eli Upfal. A trade-off between space and efficiency for routing tables. *Journal of the ACM (JACM)*, 36(3):510–530, 1989.

43 Satish B. Rao and Warren D. Smith. Approximating geometrical graphs via "spanners" and "banyans". In *Proc. 13th ACM Symposium on Theory of Computing (STOC)*, pages 540–550, 1998.

44 Jim Ruppert and Raimund Seidel. Approximating the d-dimensional complete Euclidean graph. In *Proc. 3rd Canadian Conference on Computational Geometry (CCCG)*, pages 207–210, 1991.

45 Christian Schindelhauer, Klaus Volbert, and Martin Ziegler. Geometric spanners with applications in wireless networks. *Comput. Geom.*, 36(3):197–214, 2007.

46 Michiel Smid. The weak gap property in metric spaces of bounded doubling dimension. In *Efficient Algorithms*, pages 275–289. Springer, 2009.

47 Shay Solomon. Euclidean Steiner shallow-light trees. *J. Comput. Geom.*, 6(2):113–139, 2015.

48 Subhash Suri. On some link distance problems in a simple polygon. *IEEE Trans. Robotics Autom.*, 6(1):108–113, 1990.

49 Andrew Chi-Chih Yao. On constructing minimum spanning trees in k-dimensional spaces and related problems. *SIAM J. Comput.*, 11(4):721–736, 1982.

Counting Cells of Order-k Voronoi Tessellations in $\mathbb{R}^3$ with Morse Theory

Ranita Biswas
IST Austria (Institute of Science and Technology Austria), Klosterneuburg, Austria

Sebastiano Cultrera di Montesano
IST Austria (Institute of Science and Technology Austria), Klosterneuburg, Austria

Herbert Edelsbrunner
IST Austria (Institute of Science and Technology Austria), Klosterneuburg, Austria

Morteza Saghafian
Department of Mathematical Sciences, Sharif University of Technology, Tehran, Iran

Abstract

Generalizing Lee's inductive argument for counting the cells of higher order Voronoi tessellations in $\mathbb{R}^2$ to $\mathbb{R}^3$, we get precise relations in terms of Morse theoretic quantities for piecewise constant functions on planar arrangements. Specifically, we prove that for a generic set of $n \geq 5$ points in $\mathbb{R}^3$, the number of regions in the order-k Voronoi tessellation is $N_{k-1} - \binom{k}{2}n + n$, for $1 \leq k \leq n-1$, in which N_{k-1} is the sum of Euler characteristics of these function's first $k-1$ sublevel sets. We get similar expressions for the vertices, edges, and polygons of the order-k Voronoi tessellation.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Voronoi tessellations, Delaunay mosaics, arrangements, convex polytopes, Morse theory, counting

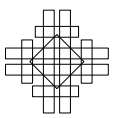
Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.16

Funding This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme, grant no. 788183, from the Wittgenstein Prize, Austrian Science Fund (FWF), grant no. Z 342-N31, and from the DFG Collaborative Research Center TRR 109, "Discretization in Geometry and Dynamics", Austrian Science Fund (FWF), grant no. I 02979-N35.

1 Introduction

The size of the Voronoi tessellation of n points in $\mathbb{R}^d$ (by which we mean the number of cells of dimension between 0 and d) is reasonably well understood, although there are still open questions. In contrast, little is known about higher order Voronoi tessellations, except in $\mathbb{R}^2$, where induction can be used to determine the size; see Lee [6]. The original motivation for the work described in this paper is the extension of the inductive argument beyond 2 dimensions. A result like in $\mathbb{R}^2$, where the size depends solely on the number of points, cannot be expected even in $\mathbb{R}^3$. Nevertheless, we report precise formulas in terms of elementary Morse theoretic concepts, such as the critical cells of piecewise constant functions on arrangements. This connection opens up the use of topological methods to counting cells and related combinatorial quantities.

Prior work. While Voronoi tessellations go back more than 100 years to the seminal work of Voronoi [10] or earlier, higher order Voronoi tessellations have been introduced only recently, by Shamos and Hoey [8] in computational geometry and by Gábor Fejes Tóth [4] in discrete geometry. Particularly important for this paper is the incremental algorithm of Lee [6], which also serves as inductive counting argument and establishes that the order-k Voronoi



tessellation of n points in $\mathbb{R}^2$, as defined in Section 2.1, has $\Theta(kn)$ vertices, edges, and regions. This implies that the first k higher order Voronoi tessellations have size $\Theta(k^2n)$. The latter bound was extended to $\Theta(k^{\lceil \frac{d+1}{2} \rceil} n^{\lfloor \frac{d+1}{2} \rfloor})$ in $\mathbb{R}^d$ by Clarkson and Shor [2]. Indeed, it is easy to give tight bounds on the total size, over all orders $1 \leq k \leq n-1$, but there are no good bounds known for individual orders beyond 2 dimensions.

To illustrate the difficulties, we mention that the size of the (order-1) Voronoi tessellation of n points in $\mathbb{R}^3$ depends not only on n but also on how the points are distributed in space. If the points are uniformly distributed within the unit cube, the expected size is $\Theta(n)$, but if the points are placed on the moment curve, then the size is $\Theta(n^2)$. On the other hand, the total size, over all orders, depends only on n and is therefore the same for both sets. This suggests that for large values of k, the uniformly distributed points have larger Voronoi tessellations than the points on the moment curve, and this has been experimentally quantified in [3].

Results. We extend the inductive approach of Lee [6] to 3 dimensions. The basis of this extension is the contractibility of the skeleta that split the regions for order $k-1$ into the pieces that combine to the regions for order k. Weaker versions of this lemma can be found in Lee [6] for $\mathbb{R}^2$ and in [3] for $\mathbb{R}^d$. The inductive approach is simplified by interpreting the tessellations in projective rather than Euclidean space. This effectively combines the order-k and the order-$(n-k)$ tessellations, with the benefit that in 2 dimensions we have precisely $(2k-1)(n-k)-(k-2)$ regions, and similar expressions for the number of edges and vertices, provided the n points be in general position. Similarly, in 3 dimensions we have precisely $N_{k-1} - \binom{k}{2}n + n$ regions, and similar expressions for the number of polygons, edges, and vertices, again with the only requirement that the points be in general position. The N_k form the connection to discrete Morse theory. Specifically, $N_k = M_1 + M_2 + \ldots + M_k$, in which M_i is the alternating sum of critical polygons of order at most i in $\binom{n}{2}$ 2-dimensional arrangements of $n-2$ lines each. For n points on the moment curve, each such arrangement has only two critical polgyons: one at order $k=1$ and the other at order $k=n-1$. Hence $N_k = k\binom{n}{2}$, for $1 \leq k \leq n-2$, and $N_{n-1} = n\binom{n}{2}$, and we get a complete description of the size but also of the combinatorial structure of the higher order Voronoi tessellations. For the general case, the determination of the N_k is however a difficult question.

Outline. Section 2 explains the background needed to appreciate this paper, which are basic geometric results on Voronoi tessellations, plane arrangements, and convex polytopes. Section 3 extends the inductive argument of Lee [6] to 3 dimensions, getting relations for the number of cells in terms of alternating sums of critical polygons. Section 4 introduces the Morse theoretic framework within which the alternating sums can be interpreted as Euler characteristics of sublevel sets of discrete Morse functions. Section 5 concludes the paper.

2 Geometric Background

We introduce order-k Voronoi tessellations and k-th Brillouin zones in d dimensions, together with an explanation of the connection to arrangements in $d+1$ dimensions. In addition, we prove that the skeleta along which the regions in the order-k tessellations split are contractible.

2.1 Voronoi Tessellations and Brillouin Zones

Let A be a finite set of points in $\mathbb{R}^d$ and write $n = \#A$ for the cardinality. For any subset $Q \subseteq A$, the *region* of Q is the set of points in $\mathbb{R}^d$ that are at least as close to the points in Q as to the points not in Q. Each such region is a d-dimensional convex polyhedron, and the

common intersection of any collection of regions, each defined by the same number of points, is either empty or a face common to all of them. We follow [6, 8] in defining the *order-k Voronoi tessellation* of A, denoted $\mathrm{Vor}_k(A)$, as the polyhedral complex whose cells are the regions defined by subsets Q of size k together with all their faces; see Figure 1, left panel. By definition, the order-0 tessellation consists of a single region, which is the entire $\mathbb{R}^d$.

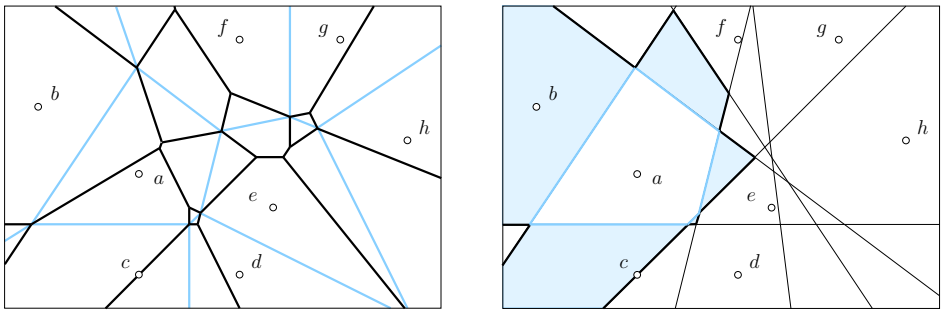


Figure 1 *Left panel:* starting with the *blue* (order-1) Voronoi tessellation of the points, we construct the order-2 Voronoi tessellation by dividing up the order-1 regions with *solid black* lines and merging them across the *blue* lines. *Right panel:* the bisectors of a and all other points divide the plane into the Brillouin zones of a. The highlighed second Brillouin zone is where a expands from the order-1 to the order-2 Voronoi tessellation; compare with *left panel*.

The set of points in $\mathbb{R}^d$ for which $a \in A$ is the k-th nearest is the *k-th Brillouin zone* of a. As illustrated in the right panel in Figure 1, this set consists of a number of regions in the arrangement formed by the bisectors of a and the other points in A. The first Brillouin zone is a convex polyhedron, and each of the other zones has the homotopy type of a sphere. Furthermore, the union of the first k zones is star-convex, with a in the kernel; see [4]. Importantly, for $k \geq 2$, every region in the k-th Brillouin zone is a d-dimensional convex polytope whose boundary can be partitioned into the *near boundary*, which is visible from a, the *far boundary*, which is not visible from a, and the *silhouette*, which separates the near and far boundaries. By convexity, the silhouette is homeomorphic to a $(d-2)$-sphere that splits the boundary into two pieces, each homeomorphic to an open $(d-1)$-ball.

2.2 Plane Arrangement

It is useful to consider the collection of d-dimensional planes in $\mathbb{R}^{d+1}$ obtained by mapping each point $a \in A$ to the affine function $\alpha\colon \mathbb{R}^d \to \mathbb{R}$ defined by $\alpha(x) = 2\langle x, a\rangle - \|a\|^2$. Note that α encodes the squared Euclidean distance from a: $\|x-a\|^2 = \|x\|^2 - \alpha(x)$. The graph of α is a (non-vertical) d-plane in $\mathbb{R}^{d+1}$. The collection of d-planes decomposes $\mathbb{R}^{d+1}$ into convex cells of dimension $0 \leq i \leq d+1$, referred to as the *arrangement* of d-planes. We call the $(d+1)$-cells *chambers*, and the d-cells *facets*. For $1 \leq k \leq n$, the *k-th level* of the arrangement is the set of points $(x, y) \in \mathbb{R}^d \times \mathbb{R}$ such that $\alpha(x) < y$ for at most $k-1$ affine maps and $\alpha(x) > y$ for at most $n-k$ affine maps. The *k-th belt* is the set of points between the k-th level and the $(k+1)$-st level.

▶ **Lemma 2.1** (From Arrangement to Tessellation). *Let A be a set of n points in $\mathbb{R}^d$, let $0 \leq k \leq n$, and recall that A defines an arrangement of n non-vertical d-planes in $\mathbb{R}^{d+1}$.*

- *There is a bijection between the regions of $\mathrm{Vor}_k(A)$ and the chambers of the k-th belt such that each region is the vertical projection of the corresponding chamber.*
- *The k-th Brillouin zone of $a \in A$ is the vertical projection of the k-th level intersected with the d-plane defined by a.*

As illustrated in Figure 2, it is convenient to take the projective view, in which we connect the levels and belts at infinity. Henceforth, this is what we mean by the k-th belt, namely the (non-projective) k-th and $(n-k)$-th belts connected at infinity, and similarly for the Voronoi tessellations, and the Brillouin zones. Figure 1 shows the non-projective concepts, and to make them projective, we would add the overlay of the order-7 and the order-6 to the left panel, and we would shade the wedge on the lower right in the right panel since it belongs to the shaded region that contains b on the left. Note that the k-th belt is the same as the $(n-k)$-th belt, for every k. Counting every cell twice, this amounts to a double-covering of the d-dimensional projective space. The main reason for adapting this view is the resulting simplification of the counting arguments and the beautification of the results.

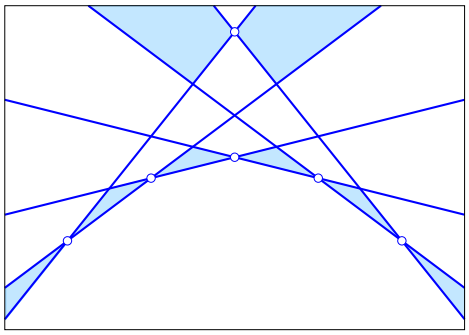


Figure 2 The first belt in the projective line arrangement consists of all chambers above exactly one line and all chambers below exactly one line. The unbounded chambers are paired up, and each pair is considered a single chamber. We thus count 6 (*light blue*) chambers in the first belt.

Another useful because simplifying assumption is that the points be in *general position*, by which we mean that any i-dimensional plane in $\mathbb{R}^d$ passes through at most $i+1$ points, and any i-dimensional sphere passes through at most $i+2$ points of A, for $0 \leq i \leq d-1$. If the points are in general position, the corresponding arrangement of d-planes in $\mathbb{R}^{d+1}$ is generic; that is: any $i+1$ d-planes intersect in a $(d-i)$-dimensional plane. This implies that any $d+2$ or more d-planes have an empty common intersection.

2.3 Convex Polytopes and Their Skeleta

Consider $n \geq d+2$ points in general position in $\mathbb{R}^d$. In the projective view, every chamber in the corresponding arrangement in $\mathbb{R}^{d+1}$ is a (bounded) convex $(d+1)$-polytope, and in the doubly-covered view, there is a second, *antipodal* copy of the polytope in the arrangement. The boundary of the chamber consists of i-dimensional cells, for $0 \leq i \leq d$, each a (convex) i-polytope itself. Because of general position, the chamber is *simple*, by which we mean that every vertex belongs to $d+1$ facets, every edge belongs to d facets, etc. It follows that every vertex belongs to $d+1$ edges, which we express by saying that the vertex has *degree* $d+1$.

By Lemma 2.1, every region in the order-k Voronoi tessellation is the vertical projection of a chamber, and by construction, the projection is *generic*, in the sense that its restriction preserves the dimension of every face in the boundary. Since the projection is in the vertical direction, it makes sense to distinguish between *lower* and *upper facets*. The k-th belt consists of chambers above k planes or below k planes, and to be consistent, we reverse upper/lower for the latter type. This will not cause any confusion since we always look at a single chamber, which we may assume is bounded and of the former type. The *lower boundary* of

such a chamber consists of all lower facets and their faces, the *upper boundary* consists of all upper facets and their faces, and the *silhouette* is the intersection of the lower and the upper boundaries. The projection of the silhouette is the boundary of the projected chamber. Although this is a generic projection of a simple $(d+1)$-polytope, it is not necessarily a simple d-polytope. Indeed, a vertex of the silhouette may be incident to i lower facets and $j = d+1-i$ upper facets, for any $1 \leq i \leq d$. Such a vertex belongs to ij $(d-1)$-cells in the silhouette. See the dodecahedron in Figure 3 as an example, which has 8 degree-3 vertices and 6 degree-4 vertices.

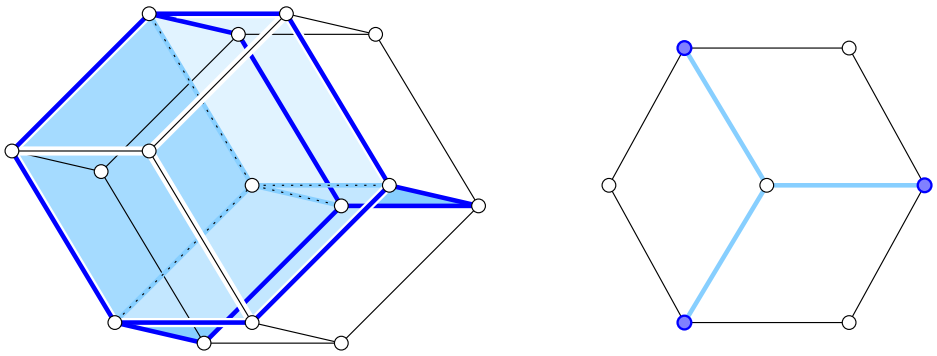


Figure 3 *Left:* projecting the 4-cube along a diagonal gives the rhombic dodecahedron in $\mathbb{R}^3$, which we see decomposed into four distorted 3-cubes. The projection of the silhouette is the boundary of the dodecahedron. The projection of the upper 2-skeleton of the 4-cube consists of the vertices, edges, and *light blue* polygons shared by the distorted 3-cubes. Its boundary is a graph in the boundary of the dodecahedron, which we highlight with *dark blue* edges. *Right:* the analogous construction one dimension lower, in which the 3-cube projects to a decomposed hexagon.

We call the cells of dimension less than d in the lower boundary minus the silhouette the *lower skeleton* of the chamber. Symmetrically, we define the *upper skeleton* of the chamber. Both skeleta are open and $(d-1)$-dimensional. The *boundary* of the lower skeleton consists of all proper faces of its $(d-1)$-cells that belong to the silhouette, and similarly for the upper skeleton. Adding their boundaries, we get the *closed lower* and *upper skeleta*. For example, the blue open trigon in the projection of the 3-cube in Figure 3 is the upper skeleton, and closed trigon is the closed upper skeleton. We will make heavy use of a topological property of the closed skeleta that does not hold for general convex polyhedra and therefore also not for chambers in general arrangements.

▶ **Lemma 2.2** (Contractible Skeleta). *Let A be a set of $n \geq d+2$ points in general position in $\mathbb{R}^d$, let $1 \leq k \leq n-1$, and consider a chamber in the k-th belt of the corresponding arrangement in $\mathbb{R}^{d+1}$. Then the closed lower skeleton of this chamber is contractible unless $k = 1$, and the closed upper skeleton is contractible, unless $k = n-1$. In the two exceptional cases, the skeleta are empty.*

Proof. We consider the lower skeleton first. Let $Q \subseteq A$ with $\#Q = k$ such that the projection of the chamber to $\mathbb{R}^d$ is the region of points that satisfy $\|x-a\| \leq \|x-b\|$ for all $a \in Q$ and all $b \in A \setminus Q$. We denote this region R, and for each $a \in Q$, we write $R_a \subseteq R$ for the subset of points for which a is the k-th nearest or, equivalently, the furthest of the points in Q. We note that R_a is convex and indeed a region of the k-th Brillouin zone of a. Assuming $k \geq 2$, the boundary of R_a can be partitioned into the near boundary, which is visible from a, the far boundary, which is not visible from a, and the silhouette, which separates the two.

Indeed, the projection of the closed lower skeleton is the union of the near boundaries of all R_a, with $a \in Q$. To prove that the closed lower skeleton is contractible, we give an explicit deformation retraction from R to the projection of the closed lower skeleton. Within R_a, the deformation retraction moves every point $x \in R_a$ straight toward a until it reaches the near boundary of R_a. This is well defined because $k \geq 2$ so that a lies outside R_a. Because R is convex and therefore contractible, the existence of this deformation retraction implies that the lower skeleton is contractible.

We consider the upper skeleton second. For each $b \in A \setminus Q$, we write $R_b \subseteq R$ for the subset of points for which b is the $(k+1)$-st nearest or, equivalently, the nearest of the points in $A \setminus Q$. Now the near boundary of R_b belongs to the boundary of R, and the projection of the closed upper skeleton is the union of far boundaries of all R_b, with $b \in A \setminus Q$. Like before, the deformation retraction moves a point $y \in R_b$ straight away from b until it hits the far boundary of R_b. This construction works for $k \leq n-2$, as claimed. ◀

For the cases in which Lemma 2.2 guarantees contractibility, we will refer to the closed skeleta as *closed* $(d-1)$*-trees* and their open versions simply as $(d-1)$*-trees*.

3 Counting in Three Dimensions

We count the cells of the Voronoi tessellations in 3 dimensions inductively, following the pattern of the 2-dimensional argument pioneered by Lee [6]. To begin, we need some understanding of 4-dimensional convex polytopes and their projections.

3.1 Chambers Projected to Regions

It is instructive to look at the 4-cube and its projection along a diagonal direction, which is known as the *rhombic dodecahedron*; see Figure 3. Each endpoint of the diagonal is incident to four 3-cubes in the boundary, and their projections form two decompositions of the rhombic dodecahedron into four distorted 3-cubes each. While the 4-cube is simple and the projection is generic, the rhombic dodecahedron is not simple: it has vertices of degree 3 and of degree 4. The two decompositions into distorted 3-cubes are by projecting the lower skeleton and the upper skeleton of the 4-cube, which by Lemma 2.2 are 2-trees. The boundary of each 2-tree is a graph in the silhouette of the 4-cube. Figure 3 shows one of these graphs, which connects some of the degree-3 vertices and all of the degree-4 vertices by dark blue edges. The other graph (not shown) uses the remaining edges in the silhouette. The two graphs are disjoint except that they share all degree-4 vertices, where they cross. This is a pattern that can be observed in general and not just in the example depicted in Figure 3.

We use the combinatorics of the 2-trees to count the cells in the order-k Voronoi tessellation, which we recall are obtained by projecting the chambers in the k-th belt of the arrangement in $\mathbb{R}^4$. It will be important to count the cells of different dimension and of different type separately. We thus use the following notation:

$$u_k = \#\text{old vertices}, \quad v_k = \#\text{mid vertices}, \quad w_k = \#\text{new vertices}, \tag{1}$$

$$d_k = \#\text{old edges}, \quad e_k = \#\text{new edges}, \tag{2}$$

$$p_k = \#\text{polygons}, \quad r_k = \#\text{regions}, \tag{3}$$

in which we call a vertex *old*, *mid*, or *new* in the order-k Voronoi tessellation if it belongs to the tessellations of orders $k-2, k-1, k$, orders $k-1, k, k+1$, or orders $k, k+1, k+2$. Similarly, we call an edge *old* or *new* in the order-k Voronoi tessellation if it belongs to the tessellations of orders $k-1, k$ or orders $k, k+1$.

3.2 Graphs and 2-Trees

Recall that the upper 2-tree of a chamber contains all vertices, edges, and polygons shared by at least two of the upper facets. This implies that the boundary of this 2-tree consists of the new edges and the mid and new vertices in the silhouette. Symmetrically, the boundary of the lower 2-tree consists of the old edges and the mid and old vertices in the silhouette. Together, the two graphs exhaust all edges and vertices, and they intersect in the mid vertices, where they cross. We call a cycle in the graph a *loop* and the cyclomatic number the *number of loops*, which for a connected graph is $\#\text{edges} - \#\text{vertices} + 1$. Let u, v, w, d, e be the numbers of old, mid, new vertices and old, new edges in the silhouette.

▶ **Lemma 3.1** (Loops in Graphs). *The boundary of the upper 2-tree is a connected graph with $\frac{1}{2}w + 1$ loops, and the boundary of the lower 2-tree is a connected graph with $\frac{1}{2}u + 1$ loops.*

Proof. It suffices to consider the boundary of the upper 2-tree. It is connected, else the 2-tree would not be contractible. It has e edges, v vertices of degree 2, and w vertices of degree 3, which implies $2e = 2v + 3w$. The number of loops is $e - (v + w) + 1 = \frac{1}{2}w + 1$. ◀

The combinatorics of the boundary is important for the combinatorics of the 2-tree, but it does not determine it. We therefore introduce a shape variable, which together with the graph determines the number of vertices, edges, and polygons in the 2-tree. The corresponding intuition will be revealed in Section 4.1. We call a polygon a *minimum* if its entire boundary belongs to the 2-tree. Otherwise, the part of the boundary in the 2-tree consists of $\mu + 1$ arcs, and we call the polygon a *maximum* if $\mu = -1$, a *non-critical polygon* if $\mu = 0$, and a *saddle* with *multiplicity* μ if $\mu \geq 1$. The shape variable of the polygon is $\#\text{edges} - \#\text{vertices} + 1$, in which we count only the faces in the 2-tree. Note that this is 1 for a minimum and maximum, 0 for a non-critical polygon, and $-\mu$ for a saddle with multiplicity μ. Taking the sum over the polygons in the upper 2-tree, we get the *characteristic* of the chamber, which we denote J. It is also defined for 2-skeleta that are not contractible, but the relation expressed in the next lemma holds only for 2-trees.

▶ **Lemma 3.2** (Size of 2-Tree). *Let A be a set of n points in general position in $\mathbb{R}^3$ and $1 \leq k \leq n - 1$. The numbers of vertices, edges, and polygons in the upper 2-tree of a chamber in the k-th belt satisfy*

$$W = \tfrac{1}{2}w - 1 + J, \tag{4}$$
$$E = \tfrac{3}{2}w - 2 + 2J, \tag{5}$$
$$P = \tfrac{3}{2}w + J, \tag{6}$$

in which w is the number of new vertices in the silhouette, and J is the characteristic of the chamber.

Proof. We first dispose of an easy case: when the 2-tree contains a maximum. Then $J = 1$ because the 2-tree contains only this one polygon and has no edges and no vertices. Furthermore, $w = 0$, so the claimed relations give $W = 0$, $E = 0$, $P = 1$, as required.

We can therefore assume that the 2-tree contains no maximum, but there may be minima and saddles beside the non-critical polygons. Consider the graph formed by the edges and vertices in the 2-tree, to which we add the w new vertices in the silhouette so that each edge has both endpoints. For each minimum in the 2-tree, this graph contains a loop, and for each saddle with multiplicity μ, the graph contains μ extra components. The characteristic is $J = \#\text{loops} - \#\text{components} + 1$, and the number of edges is

$$E = W + w + \#\text{loops} - \#\text{components} = W + w - 1 + J. \tag{7}$$

We also have $2E = 4W + w$, which combined with (7) implies (4) and (5). To prove (6), we recall that the closed 2-tree is contractible, which implies that its Euler characteristic satisfies $(W - E + P) + (v + w - e) = 1$. Substituting $E = 2W + \frac{1}{2}w$ and $e = v + \frac{3}{2}w$ implies $P = W + w + 1 = \frac{3}{2}w + J$, as required. ◀

For counting purposes, we write J_{k+1} for the sum of characteristics of the chambers in the k-th belt, for $0 \leq k \leq n-1$. By Lemma 2.2, the upper 2-skeleton of every chamber is contractible and therefore a 2-tree for $1 \leq k \leq n-2$. For $k = 0$, there is a single chamber whose upper boundary projects to the entire first Voronoi tessellation. Its upper 2-skeleton is therefore not a 2-tree, but its characteristics is still defined, namely the number of polygons in $\mathrm{Vor}_1(A)$.

3.3 Induction

We count the vertices, edges, polygons, and regions of the Voronoi tessellations inductively, beginning with order $k = 1$.

▶ **Lemma 3.3** (Induction Basis in $\mathbb{R}^3$)**.** *The order-*1 *Voronoi tessellation of $n \geq 5$ points in general position in $\mathbb{R}^3$ has*

$$w_1 = J_1 - n \quad \textit{vertices}, \tag{8}$$
$$e_1 = 2J_1 - 2n \quad \textit{edges}, \tag{9}$$
$$p_1 = J_1 \quad \textit{polygons}, \tag{10}$$
$$r_1 = n \quad \textit{regions}. \tag{11}$$

Proof. We have $p_1 = J_1$ by definition, and $r_1 = n$ because the order-1 Voronoi tessellation has one region for each point. To get the relations for the vertices and edges, we note that the order-1 Voronoi tessellation is a polyhedral complex that decomposes the 3-sphere, so that Euler's formula implies $w_1 - e_1 + p_1 - r_1 = 0$. By assumption of general position, every vertex belongs to 4 edges, which gives $4w_1 - 2e_1 = 0$. Combining these two relations, we can express w_1 and e_1 in terms of p_1 and r_1 and therefore in terms of J_1 and n, as stated. ◀

When we go from the order-$(k-1)$ to the order-k Voronoi tessellation, we see some cells die, some cells age, and some cells get born according to relations (4), (5), (6).

▶ **Lemma 3.4** (Induction Step in $\mathbb{R}^3$)**.** *The numbers of old, mid, new vertices, old and new edges, polygons, and regions of the order-k Voronoi tessellation of n points in general position in $\mathbb{R}^3$ satisfy*

$$u_k = v_{k-1}, \tag{12}$$
$$v_k = w_{k-1}, \tag{13}$$
$$w_k = 2w_{k-1} - r_{k-1} + J_k, \tag{14}$$
$$d_k = e_{k-1}, \tag{15}$$
$$e_k = 6w_{k-1} - 2r_{k-1} + 2J_k, \tag{16}$$
$$p_k = 6w_{k-1} + J_k, \tag{17}$$
$$r_k = w_{k-1} - v_{k-1} + r_{k-1}, \tag{18}$$

for $2 \leq k \leq n-1$.

Proof. Rules (12), (13), (15) express aging. By assumption of general position, each new vertex of $\mathrm{Vor}_{k-1}(A)$ belongs to four regions, so we get rule (14) from (4), rule (16) from (5), and rule (17) from (6). To get rule (18), we note that each of the $u_k + w_k$ old and new vertices has degree 4, and each of the v_k mid vertices has degree 8, again by assumption of general position. Hence $2(d_k + e_k) = 4(u_k + 2v_k + w_k)$. Plugging this into the Euler formula for the 3-sphere, we get $r_k = (u_k + v_k + w_k) - (d_k + e_k) + p_k = p_k - (u_k + 3v_k + w_k)$, which implies (18). ◀

These rules can be used to find expressions for the cells in the Voronoi tessellations. Recall that J_k is the alternating sum of critical polygons of order k; see text following the proof of Lemma 3.2. It will be convenient to write $M_k = J_1 + J_2 + \ldots + J_k$ and $N_k = M_1 + M_2 + \ldots + M_k = kJ_1 + (k-1)J_2 + \ldots + J_k$, and to set $J_k = M_k = N_k = 0$ for $k \leq 0$.

▶ **Theorem 3.5** (Size of Order-k Voronoi Tessellations in $\mathbb{R}^3$)**.** *For $1 \leq k \leq n-1$, the order-k Voronoi tessellation of $n \geq 5$ points in $\mathbb{R}^3$ has*

$$u_k \leq N_{k-2} - \tbinom{k-1}{2} n \quad \textit{old vertices}, \tag{19}$$
$$v_k \leq N_{k-1} - \tbinom{k}{2} n \quad \textit{mid vertices}, \tag{20}$$
$$w_k \leq N_k - \tbinom{k+1}{2} n \quad \textit{new vertices}, \tag{21}$$
$$d_k \leq 2N_{k-1} + 2N_{k-2} - 2(k-1)^2 n \quad \textit{old edges}, \tag{22}$$
$$e_k \leq 2N_k + 2N_{k-1} - 2k^2 n \quad \textit{new edges}, \tag{23}$$
$$p_k \leq 6N_{k-1} + J_k - 6\tbinom{k}{2} n \quad \textit{polygons}, \tag{24}$$
$$r_k \leq N_{k-1} - \tbinom{k}{2} n + n \quad \textit{regions}, \tag{25}$$

with equality in all seven cases if the points are in general position.

Proof. Let A be a set of $n \geq 5$ points in $\mathbb{R}^3$. Whenever A is not in general position, we can perturb it into general position without losing any vertex, edge, polygon, or region in any of its Voronoi tessellations. We can therefore assume without loss of generality that A is in general position and prove that in this case the seven claimed inequalities are equations.

For $k = 1$, we have $u_1 = v_1 = 0$, $w_1 = J_1 - n$, $d_1 = 0$, $e_1 = 2J_1 - 2n$, $p_1 = J_1$, and $r_1 = n$, which agrees with Lemma 3.3. Assuming the relations are correct for index $k-1$, we use Lemma 3.4 to prove that they are correct for k. (19), (20), (22) follow straightforwardly from (12), (13), (15) and (21), (23). To see (21), (23), (24), (25), we use (14), (16), (17), (18) to compute

$$w_k = [2N_{k-1} - N_{k-2} + J_k] - [2\tbinom{k}{2} - \tbinom{k-1}{2} + 1]n = N_k - \tbinom{k+1}{2} n, \tag{26}$$
$$e_k = [6N_{k-1} - 2N_{k-2} + 2J_k] - [6\tbinom{k}{2} - 2\tbinom{k-1}{2} + 2]n = 2N_k + 2N_{k-1} - 2k^2 n, \tag{27}$$
$$p_k = [6N_{k-1} + J_k] - 6\tbinom{k}{2} n, \tag{28}$$
$$r_k = [N_{k-1} - N_{k-2} + N_{k-2}] - [\tbinom{k}{2} - \tbinom{k-1}{2} + \tbinom{k-1}{2} - 1]n = N_{k-1} - \tbinom{k}{2} n + n, \tag{29}$$

as claimed. ◀

3.4 Relations of Symmetry

By projective interpretation, the order-k Voronoi tessellation is also the order-$(n-k)$ Voronoi tessellation. Indeed, the only difference between the two is that upper and lower facets switch, and so do old and new vertices and old and new edges. We state these relations formally and use them to derive a similar relation for the characteristics of the belts.

▶ **Theorem 3.6** (Symmetry Relations). *The numbers of old, mid, new vertices, old and new edges, polygons, regions, and the characteristics of the order-k Voronoi tessellations of $n \geq 5$ points in general position in $\mathbb{R}^3$ satisfy $u_k = w_{n-k}$, $v_k = v_{n-k}$, $w_k = u_{n-k}$, $d_k = e_{n-k}$, $e_k = d_{n-k}$, $p_k = p_{n-k}$, $r_k = r_{n-k}$, $J_k = J_{n-k}$, for $1 \leq k \leq n-1$.*

Proof. The symmetry relations for the vertices, edges, polygons, and regions follow from the symmetry of the projective definition of order-k Voronoi tessellations. To see the relation for the characteristic, we note that $\binom{k+1}{2} - 2\binom{k}{2} + \binom{k-1}{2} = 1$ and that $N_k - 2N_{k-1} + N_{k-2} = M_k - M_{k-1} = J_k$, for $1 \leq k \leq n-1$. Using $r_k = N_{k-1} - \binom{k}{2}n + n$ from Theorem 3.5 and the symmetry relation for the regions, we get

$$0 = [r_{k+1} - 2r_k + r_{k-1}] - [r_{n-k-1} - 2r_{n-k} + r_{n-k+1}] = [J_k - n] - [J_{n-k} - n]. \tag{30}$$

Simplifying (30), we get the claimed symmetry relation for the characteristic. ◀

To provide examples, we list the number of cells and characteristics of belts for two sets of six points each in Table 1. Observe the symmetry in the columns as predicted by Theorem 3.6, and note that the numbers given for the moment curve example are consistent with the expressions given in Section 1.

Table 1 Numbers of vertices, edges, polygons, and regions for six points on the moment curve in $\mathbb{R}^3$ in the *upper table*, and of the six points of the double suspended tetrahedron in the *lower table*.

k	u_k	v_k	w_k	d_k	e_k	p_k	r_k	J_k	M_k	N_k
1	0	0	9	0	18	15	6	15	15	15
2	0	9	12	18	42	54	15	0	15	30
3	9	12	9	42	42	72	18	0	15	45
4	12	9	0	42	18	54	15	0	15	60
5	9	0	0	18	0	15	6	15	30	90
$\sum$	30	30	30	120	120	210	60	30	90	240

k	u_k	v_k	w_k	d_k	e_k	p_k	r_k	J_k	M_k	N_k
1	0	0	8	0	16	14	6	14	14	14
2	0	8	14	16	44	52	14	4	18	32
3	8	14	8	44	44	78	20	-6	12	44
4	14	8	0	44	16	52	14	4	16	60
5	8	0	0	16	0	14	6	14	30	90
$\sum$	30	30	30	120	120	210	60	30	90	240

4 A Morse Theoretic Perspective

The J_k, M_k, N_k have alternative interpretations in terms of sublevel sets of discrete Morse functions. As we will see, these functions are closer in spirit to the discrete Morse theory introduced by Banchoff [1] than the more popular version developed by Forman [5]. It is convenient to adopt the language of great-circle arrangements in the sphere instead of doubly-covered projective line arrangements in the plane, which are of course equivalent.

4.1 Arrangements of Great-Circles

Let A be a set of n points in general position, let a, b, c be three different points in A, and recall that they correspond to affine functions $\alpha, \beta, \gamma \colon \mathbb{R}^3 \to \mathbb{R}$. We are interested in $\alpha(x) = \beta(x)$, which describes a plane in $\mathbb{R}^3$, in $\gamma(x) = \alpha(x) = \beta(x)$, which describes a line, and in $\gamma(x) \leq \alpha(x) = \beta(x)$, which describes a half-plane; see Figure 4. In our spherical view, the plane becomes a sphere, denoted $S_{a,b}$, and for each point c, we get a hemi-sphere,

$H_c \subseteq S_{a,b}$. The $n-2$ lines decomposing the plane correspond to the same number of great-circles that decompose the sphere into vertices, edges, and (spherical) polygons, which appear in antipodal pairs. The main concept in this section is the function

$$f_{a,b}(\varphi) = 1 + \#\{c \in A \setminus \{a,b\} \mid \varphi \subseteq H_c\}, \tag{31}$$

in which φ is a polygon in the great-circle arrangement on $S_{a,b}$. For completeness, we define $f_{a,b}$ for an edge or a vertex equal to the smallest value of any polygon incident to the edge or vertex. Write $\bar{\varphi}$ for the antipodal polygon of φ, and call a polygon, ψ, a *neighbor* of φ if the two share an edge. Then we have

$$f_{a,b}(\varphi) = f_{a,b}(\psi) \pm 1, \tag{32}$$
$$f_{a,b}(\varphi) = n - f_{a,b}(\bar{\varphi}), \tag{33}$$

for all neighbors ψ of φ. We get (32) by assumption of general position, and (33) by symmetry. Consistent with Section 3, we call φ a *minimum* if $f_{a,b}(\varphi) = f_{a,b}(\psi) - 1$ and a *maximum*

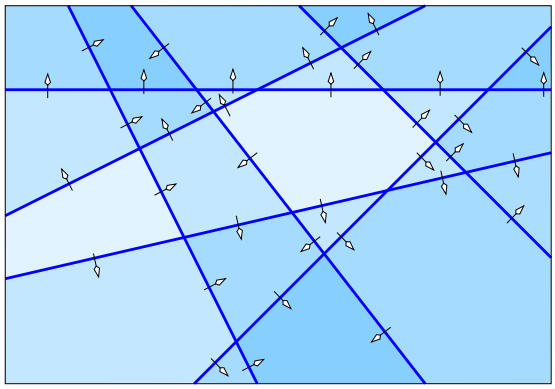


Figure 4 An arrangement of lines in which the shading indicates the coverage with half-planes. The arrows go from smaller to larger coverage. Considering only polygons that are fully contained inside the box, we see one minimum, one simple saddle, and seven non-critical polygons.

if $f_{a,b}(\varphi) = f_{a,b}(\psi) + 1$ for all neighbors ψ of φ. Otherwise, the function values of the cyclically ordered neighbors change $2(\mu+1) \geq 2$ times between $f_{a,b}(\varphi) \pm 1$. For $\mu = 0$, φ is a *non-critical polygon*, and for $\mu \geq 1$, it is a *saddle* with *multiplicity* μ.

4.2 Sublevel Sets and Euler Characteristics

A common feature in Morse theoretic studies is the occurrence of sublevel sets and their relations. For each k, the sublevel set $f_{a,b}^{-1}(-\infty, k]$ is a closed subset of $S_{a,b}$, with well-defined *Betti numbers*, $\beta_p(k)$, and *Euler characteristic*, $\chi(k) = \beta_0(k) - \beta_1(k) + \beta_2(k)$. Define $\text{index}(\varphi) = 0, 1, 2$ if φ is a minimum, saddle, maximum. In the discrete setting at hand, it is not difficult to prove an analog of the Euler–Poincaré Theorem, which implies that $\chi(k)$ is the sum of $(-1)^{\text{index}(\varphi)}$, over all critical polygons that satisfy $f_{a,b}(\varphi) \leq k$, in which a saddle with multiplicity μ is counted as μ simple saddles. We define $J_k(a,b) = \chi(k) - \chi(k-1)$, $M_k(a,b) = \chi(k)$, and $N_k(a,b) = \chi(1) + \chi(2) + \ldots + \chi(k)$. The connection to the previous discussion and, in particular, Theorem 3.5 should be clear, namely

$$J_k = \sum\nolimits_{a,b\in A} J_k(a,b), \tag{34}$$

$$M_k = \sum\nolimits_{a,b\in A} M_k(a,b), \tag{35}$$

$$N_k = \sum\nolimits_{a,b\in A} N_k(a,b), \tag{36}$$

in which the sums are over all unordered pairs in A. In other words, we have a piecewise constant function on the disjoint union of $\binom{n}{2}$ spheres, $f : \mathbb{S}^2 \sqcup \mathbb{S}^2 \sqcup \ldots \sqcup \mathbb{S}^2 \to [1, n-1]$, whose components are the functions $f_{a,b}$, such that M_k is the Euler characteristic of $f^{-1}(-\infty, k]$, and J_k is the increment from $k-1$ to k. Equivalently, J_k is the alternating sum of critical polygons in $f^{-1}(k)$. We note that this interpretation implies a strengthening of the relation $J_k = J_{n-k}$ in Theorem 3.6: $J_k(a,b) = J_{n-k}(a,b)$, for all pairs $a \neq b$, because antipodal polygons have the same contribution to the sum.

4.3 Relations for Sums

Observe that a generic arrangement of n 3-dimensional great-spheres in $\mathbb{S}^4$ has $2\binom{n}{4}$ vertices, $8\binom{n}{4}$ edges, $12\binom{n}{4} + 2\binom{n}{2}$ polygons, $8\binom{n}{4} + 4\binom{n}{2}$ facets, and $2\binom{n}{4} + 2\binom{n}{2} + 2$ chambers. This implies relations on the number of cells in the Voronoi tessellations.

▶ **Theorem 4.1** (Sum Relations). *The new vertices, new edges, polygons, and regions of the Voronoi tessellations of $n \geq 5$ points in general position in $\mathbb{R}^3$ satisfy*

$$\sum\nolimits_{k=1}^{n-1} w_k = 2\tbinom{n}{4}, \tag{37}$$

$$\sum\nolimits_{k=1}^{n-1} e_k = 8\tbinom{n}{4}, \tag{38}$$

$$\sum\nolimits_{k=1}^{n-1} p_k = 12\tbinom{n}{4} + 2\tbinom{n}{2}, \tag{39}$$

$$\sum\nolimits_{k=1}^{n-1} r_k = 2\tbinom{n}{4} + 2\tbinom{n}{2}. \tag{40}$$

Similarly, the characteristics of the belts, their cumulative sums, and the cumulative sums of those satisfy

$$\sum\nolimits_{k=1}^{n-1} J_k = 2\tbinom{n}{2}, \tag{41}$$

$$\sum\nolimits_{k=1}^{n-1} M_k = n\tbinom{n}{2}, \tag{42}$$

$$\sum\nolimits_{k=1}^{n-1} N_k = \left[\tbinom{n}{2} + 1\right] \tbinom{n}{2}. \tag{43}$$

Proof. The new vertices, new edges, polygons, and regions of the order-k Voronoi tessellations, for $1 \leq k \leq n-1$, are in bijection with the vertices, edges, polygons, and chambers of the arrangement in $\mathbb{S}^4$. Exceptions are the chamber below and above all great-spheres, which are not covered by the tessellations. This implies (37), (38), (39), (40).

We get (41) because the sum of the J_k is equal to the sum of the Euler characteristics of $\binom{n}{2}$ 2-spheres. To see (42), we note that for a pair of antipodal critical polygons, we have $[n - f(\varphi)] + [n - f(\bar{\varphi})] = n$ because $f(\varphi) + f(\bar{\varphi}) = n$ by (33). The contribution of the pair to the sum of the M_k is therefore n if $\varphi, \bar{\varphi}$ are a minimum and a maximum, and $-\mu n$ if $\varphi, \bar{\varphi}$ are saddles with multiplicity μ. The contributions cancel, except for one minimum/maximum pair per 2-sphere, and since there are $\binom{n}{2}$ 2-spheres, the sum of the M_k is $n\binom{n}{2}$. To prove (43), we use (40) and (25), which for points in general position is an equality:

$$\sum\nolimits_{k=1}^{n-1} N_{k-1} = \sum\nolimits_{k=1}^{n-1} r_k + n \sum\nolimits_{k=1}^{n-1} \tbinom{k}{2} - n \sum\nolimits_{k=1}^{n-1} 1 = 2\tbinom{n}{4} + n\tbinom{n}{3}. \tag{44}$$

By index transformation, the left-hand side is $\sum_{k=1}^{n-2} N_k$, and by straightforward calculations, the right-hand side is $\binom{n}{2}\binom{n-1}{2}$. Adding $N_{n-1} = n\binom{n}{2}$ from (42) on both sides implies the claimed relation. ◀

Observe that the relations in Theorem 4.1 are consistent with the column sums in Table 1, which are the same for the two sets of six points each.

4.4 Relation for Individual Sphere

The proofs of (41) and (42) show that each sphere contributes the same amount, namely 2 to $\sum J_k$ and n to $\sum M_k$. The proof of (43) does not show the same for $\sum N_k$, but it is still true, that is: each sphere contributes the same amount to the sum of the N_k.

▶ **Theorem 4.2** (Stronger Sum Relation). *Let A be a set of $n \geq 5$ points in general position in $\mathbb{R}^3$. Then $\sum_{k=1}^{n-1} N_k(a,b) = 1 + \binom{n}{2}$ for any two points $a \neq b$ in A.*

Proof. Fix $a, b \in A$ and consider the arrangement of $n-2$ great-circles on $S_{a,b}$. For convenience, we write J_k, M_k, N_k, and f for $J_k(a,b)$, $M_k(a,b)$, $N_k(a,b)$, and $f_{a,b}$ throughout this proof. The goal is to show $\sum_{k=1}^{n-1} N_k = 1 + \binom{n}{2}$, but we already have $N_{n-1} = n$ from (42) and $N_{n-2} = N_{n-1} - M_{n-1} = n-2$ from (41) and (42). Indeed, the proofs of (41) and (42) imply the stronger relations for individual spheres. Therefore it suffices to prove

$$X = \sum_{k=1}^{n-3} N_k = 1 + \binom{n}{2} - n - (n-2) = \binom{n-2}{2}. \tag{45}$$

We rewrite the sum in terms of the J_k. A polygon contributes to J_k only if $f(\psi) = k$, and this contribution depends on the cyclic sequence of neighboring polygons. Distinguishing between polygons φ with $f(\varphi) = k \pm 1$, the contribution to J_k is 1 minus half the number of alternations between these two types along the cycle. We therefore write $J_k = p_k - \frac{1}{2}t_k$, in which p_k is the number of polygons $\psi \subseteq S_{a,b}$ with $f(\psi) = k$, and t_k is the number of triplets of polygons (φ, ψ, ϱ) with $f(\varphi)+1 = f(\psi) = f(\varrho)-1 = k$ that share a common vertex. This vertex is where the type of neighboring polygons changes. We call such an ordered triplet of polygons a *short increasing path*. Using $N_k = kJ_1 + (k-1)J_2 + \ldots + J_k$, we get

$$\begin{aligned} X &= N_1 + N_2 + \ldots + N_{n-3} && (46)\\ &= \tbinom{n-2}{2}J_1 + \tbinom{n-3}{2}J_2 + \ldots + \tbinom{2}{2}J_{n-3} && (47)\\ &= \sum_{k=1}^{n-3}\tbinom{n-k-1}{2}p_k - \tfrac{1}{2}\sum_{k=1}^{n-3}\tbinom{n-k-1}{2}t_k. && (48) \end{aligned}$$

Write Y and Z for the two sums in (48) so that $X = Y - \frac{1}{2}Z$. Observe that Y is the number of triplets (ψ, H, H'), in which ψ is a polygon and $H \neq H'$ are two hemi-spheres that both do not contain ψ. Indeed, if $f(\psi) = k$, then there are $(n-2)-(k-1) = n-k-1$ hemi-spheres that do not contain ψ. Similarly, Z is the number of triplets $((\varphi, \psi, \varrho), H, H')$, in which (φ, ψ, ϱ) is a short increasing path and $H \neq H'$ are two hemi-spheres that both do not contain ψ. We call (ψ, H, H') a *captured polygon* and $((\varphi, \psi, \varrho), H, H')$ a *captured short increasing path*. In words, $X = Y - \frac{1}{2}Z$ counts the captured polygons but subtracts half the captured short increasing paths.

Now fix two hemi-spheres, H and H', and consider their *lune*, which is the closure of the points neither contained in H nor in H'; see Figure 5. We are interested in the portion of the arrangement of great-circles in this lune. Each vertex of this portion lies either in the interior or on the boundary. For each interior vertex, we get two captured short increasing paths, and

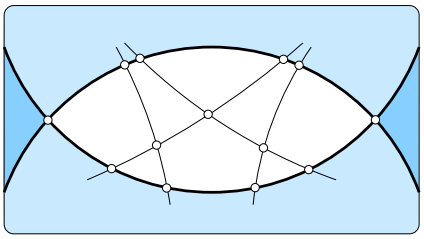


Figure 5 A lune identifies a portion of the arrangement of great-circles. Within this arrangement, we count polygons and short paths whose middle polygons are in the lune. We have two short increasing paths for each interior vertex and one for each boundary vertex, unless it is a corner of the lune, in which case we get no such path.

for each boundary vertex, we get one such path, except for the two corners of the lune for which we get no such path. Writing V_{int} and V_{bd} for the numbers of vertices in the interior and on the boundary, the number of captured short increasing paths is $2V_{int} + V_{bd} - 2$. The number of captured polygons is just the number of polygons in the lune, which we denote P. Ordering the lunes arbitrarily and writing X_i, Y_i, Z_i for the contributions of the lune defined by H and H', we have

$$X_i = Y_i - \tfrac{1}{2}Z_i = P - \tfrac{1}{2}[2V_{int} + V_{bd} - 2]. \tag{49}$$

But $P = V_{int} + \frac{1}{2}V_{bd}$, which is easy to prove by adding the great-circles one at a time and counting how many vertices and polygons a great-circle adds to the portion of the arrangement in the lune. Hence $X_i = 1$. This implies $X = \sum_i X_i = \binom{n-2}{2}$ and therefore $\sum_{k=1}^{n-1} N_k = 1 + \binom{n}{2}$, as required. ◀

5 Discussion

The main contributions of this paper are an extension of the inductive argument for counting cells in order-k Voronoi tessellations from 2 to 3 dimensions, and the Morse theoretic perspective in which the number of cells are interpreted as alternating sums of critical polygons of 2-dimensional discrete Morse functions. Alternatively, we can state the results in terms of k-sets of n points on the unit 3-sphere or, more generally, for n points in convex position in $\mathbb{R}^4$. There are connections between the alternating sums and the persistent homology of the discrete Morse functions, which may be interesting to develop in the future. There are a number of open questions this work raises:

- Can the Morse theoretic interpretation of higher order Voronoi tessellations be used to prove new upper and lower bounds on the maximum size of the order-k Voronoi tessellation of n points in $\mathbb{R}^3$? In particular, can we prove an upper bound asymptotically smaller than k^2n^2 or a lower bound asymptotically larger than k^2n; see [2, 7]?
- Can the inductive approach be generalized to sets beyond 3 dimensions? Clearly yes, but how much more complicated does it get? Can we still hope for relations that involve only one independent variable, like N_k in $\mathbb{R}^3$, or do we get extra independent variables?
- The regions in the order-k Voronoi tessellation correspond to k-sets of points on a sphere in $\mathbb{R}^4$. Can the inductive approach be extended to points not necessarily in convex position? Proving bounds on the maximum number of k-sets in this more general setting is a notoriously difficult combinatorial problem, and any advance would be exciting [9].

References

1 Thomas Banchoff. Critical points and curvature for embedded polyhedra. *Journal of Differential Geometry*, 1:245–256, 1967.

2 Kenneth L. Clarkson and Peter W. Shor. Applications of random sampling in computational geometry, II. *Discrete & Computational Geometry*, 4:387–421, 1989.

3 Herbert Edelsbrunner and Georg Osang. A simple algorithm for higher-order Delaunay mosaics and alpha shapes. *Journal of Geometry*, 2021. (accepted).

4 Gábor Fejes Tóth. Multiple packing and covering of the plane with circles. *Acta Mathematica Academiae Scientiarum Hungaricae*, 27:135–140, 1976.

5 Robin Forman. Morse theory for cell complexes. *Advances in Mathematics*, 134(1):90–145, 1998.

6 Der-Tsai Lee. On k-nearest neighbor Voronoi diagrams in the plane. *IEEE Transactions on Computers*, 31(6):478–487, 1982.

7 Ketan Mulmuley. Output sensitive construction of levels and Voronoi diagrams in R^d of order 1 to k. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, STOC'90, pages 322–330, 1990.

8 Michael I. Shamos and Dan Hoey. Closest-point problems. In *Proceedings of the 16th Annual Symposium on Foundations of Computer Science*, SFCS'75, pages 151–162, 1975.

9 Micha Sharir, Shakhar Smorodinsky, and Gábor Tardos. An improved bound for k-sets in three dimensions. *Discrete & Computational Geometry*, 26:195–204, 2001.

10 Georges Voronoi. Nouvelles applications des paramètres continus à la théorie des formes quadratiques. Deuxième mémoire: recherches sur les parallélo̊edres primitifs. *Journal für die reine und angewandte Mathematik*, 134:198–287, 1908.

Tracing Isomanifolds in $\mathbb{R}^d$ in Time Polynomial in d Using Coxeter-Freudenthal-Kuhn Triangulations

Jean-Daniel Boissonnat
Université Côte d'Azur, Inria, Sophia-Antipolis, France

Siargey Kachanovich
Université Côte d'Azur, Inria, Sophia-Antipolis, France

Mathijs Wintraecken
IST Austria (Institute of Science and Technology Austria), Klosterneuburg, Austria

Abstract

Isomanifolds are the generalization of isosurfaces to arbitrary dimension and codimension, i.e. submanifolds of $\mathbb{R}^d$ defined as the zero set of some multivariate multivalued smooth function $f : \mathbb{R}^d \to \mathbb{R}^{d-n}$, where n is the intrinsic dimension of the manifold. A natural way to approximate a smooth isomanifold $\mathcal{M}$ is to consider its Piecewise-Linear (PL) approximation $\hat{\mathcal{M}}$ based on a triangulation $\mathcal{T}$ of the ambient space $\mathbb{R}^d$. In this paper, we describe a simple algorithm to trace isomanifolds from a given starting point. The algorithm works for arbitrary dimensions n and d, and any precision D. Our main result is that, when f (or $\mathcal{M}$) has bounded complexity, the complexity of the algorithm is polynomial in d and $\delta = 1/D$ (and unavoidably exponential in n). Since it is known that for $\delta = \Omega(d^{2.5})$, $\hat{\mathcal{M}}$ is $O(D^2)$-close and isotopic to $\mathcal{M}$, our algorithm produces a faithful PL-approximation of isomanifolds of bounded complexity in time polynomial in d. Combining this algorithm with dimensionality reduction techniques, the dependency on d in the size of $\hat{\mathcal{M}}$ can be completely removed with high probability. We also show that the algorithm can handle isomanifolds with boundary and, more generally, isostratifolds. The algorithm for isomanifolds with boundary has been implemented and experimental results are reported, showing that it is practical and can handle cases that are far ahead of the state-of-the-art.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Coxeter triangulation, Kuhn triangulation, permutahedron, PL-approximations, isomanifolds/solution manifolds/isosurfacing

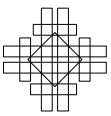
Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.17

Related Version *Full Version*: `https://hal.archives-ouvertes.fr/hal-03006663`

Funding The research leading to these results has received funding from the European Research Council (ERC) under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement No. 339025 GUDHI (Algorithmic Foundations of Geometry Understanding in Higher Dimensions).
Jean-Daniel Boissonnat: Supported by the French government, through the 3IA Côte d'Azur Investments in the Future project managed by the National Research Agency (ANR) with the reference number ANR-19-P3IA-0002.
Mathijs Wintraecken: Supported by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 754411.

Acknowledgements We thank Dominique Attali, Guilherme de Fonseca, Arijit Ghosh, Vincent Pilaud and Aurélien Alvarez for their comments and suggestions. We also acknowledge the reviewers.

37th International Symposium on Computational Geometry (SoCG 2021).
Editors: Kevin Buchin and Éric Colin de Verdière; Article No. 17; pp. 17:1–17:16



Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Given a surface represented in $\mathbb{R}^3$ as the zero set of a function $f : \mathbb{R}^3 \to \mathbb{R}$, the goal of isosurfacing is to find a piecewise linear (PL) approximation of the surface. This question naturally extends to isomanifolds of higher dimensions and codimensions defined as the zero set of multivariate multivalued smooth functions $f : \mathbb{R}^d \to \mathbb{R}^{d-n}$. Isosurfaces play a crucial role in medical imaging, computer graphics and geometry processing [22]. Higher dimensional isomanifolds are also of fundamental importance in many fields like statistics [10], dynamical systems [25], econometrics, or mechanics [22].

State-of-the-art. The most widely used algorithm to trace isomanifolds is the Marching Cube (MC) algorithm and its numerous variants [17, 27]. The MC algorithm uses a cubical grid to tesselate the ambient space. In many applications in 3-dimensions, the ambient space is decomposed into unstructured tetrahedral meshes, which led to the development of a variant of the MC algorithm named the Marching Tetrahedra algorithm. In higher dimensions, any tessellation of the ambient space has a complexity that depends exponentially on the ambient dimension. Hence a key to extending marching algorithms to higher dimensions is to circumvent the curse of dimensionality by using an *implicit* representation of the ambient tessellation. This is impossible for general triangulations but easy to do if one uses a grid. However, using a grid has other drawbacks and is not sufficient to break the exponential barrier. The reason for this is that the number of configurations inside a cubical cell grows exponentially with the dimension [27].

Hence the most promising approach seems to be to subdivide the ambient space $\mathbb{R}^d$ using a highly regular triangulation such as the Freudenthal-Kuhn triangulation. Some early work along this direction has been published in Applied Mathematics [2, 15, 25], and a slightly more recent paper by Dobkin et al. [13] attracted the interest of the Computer Graphics community to the related Coxeter triangulations. Dobkin et al. however only considered the case of curves ($n = 1$). The most advanced work we are aware of is due to Min [21]. Min's method uses the Freudenthal-Kuhn triangulation over a dyadic grid of $\mathbb{R}^d$ and applies to isomanifolds of any dimension and codimension. The time complexity of Min's method is, with our notations, $O(\delta^n \log \delta)$, where $\delta = 1/D$ and D is the maximal diameter of the simplices. The ambient dimension d is a constant hidden in the big O. The fact that the exponent of δ is the intrinsic dimension n, and not the ambient dimension d is a clear improvement over earlier methods. However, although not explicitly analysed by Min, the complexity in d remains exponential, and the method seems to be limited to small ambient dimensions. Experimental results are only reported in 3, and 4D.

Contributions. This paper discusses an efficient algorithm to compute a PL-approximation of isomanifolds. We extend the work of Dobkin et al. [13] and describe a simple algorithm to trace an n-dimensional isomanifold $\mathcal{M}$ of $\mathbb{R}^d$ for arbitrary n and d. Our algorithm uses any triangulation of a family of regular triangulations of $\mathbb{R}^d$ that includes the Coxeter and the Freudenthal-Kuhn triangulations. Contrary to Min [21], our results are obtained with a uniform triangulation leading to a very simple algorithm. Key to our results, is a data structure that can implicitly store the full facial structure of such triangulations. The data structure is very compact and allows to retrieve the faces or the cofaces of a simplex of any dimension in an output sensitive way. Using this data structure, one can trace a connected submanifold of $\mathbb{R}^d$, starting from a given initial point on the manifold (Section 3). Our algorithm produces a PL-approximation of size polynomial in d and $\delta = 1/D$, and exponential in n. The complexity of the algorithm is also polynomial in d, and δ, and exponential in n.

Moreover, by taking δ large enough, the PL-approximation output by the algorithm is a faithful approximation of the isomanifold. Specifically, as shown in the full version of [9] and recalled in Section 2.2, if we take $\delta = \Omega(d^{2.5})$, the PL-approximation $\hat{\mathcal{M}}$ is $O(D^2)$-close and isotopic to the isomanifold. Here the constants in the O depend on f and its derivatives. Hence, our algorithm constructs geometrically close and topologically correct PL-approximation of isomanifolds of bounded complexity in polynomial time.

Our algorithm can be extended in several directions. First, the dependency on d in the size of $\hat{\mathcal{M}}$ can be completely removed by combining our algorithm with dimensionality reduction (Section 3.4). We can also extend the algorithm to the case of isomanifolds with boundary and, more generally, to stratifolds (Section 3.5).

The algorithm has been implemented. In Section 4, we report on experimental results which show that the algorithm is practical and can handle cases that are far 16 ahead of the state-of-the-art. We also present an application in Algebraic Geometry that was used to verify a conjecture on projective varieties defined by polynomial equations in the complex projective plane. Following numerous experiments on various projective varieties, the conjecture was ultimately proved by Alvarez and Deroin [4].

The approximation of a manifold that is the zero set of a function is an example of the more general question of how to triangulate a manifold which has a long history in Mathematics. In particular, Whitney [28] introduced a construction that has some similarity with the present algorithm (see [7]). A major difference though is that topological guarantees can only be obtained if some intricate perturbations of the ambient triangulation are performed (Section 5). These techniques are at the moment incompatible with polynomial complexity.

2 Background

2.1 Permutahedral representation of CFK-triangulations

In this section, we give the most important definitions and basic properties of Coxeter and Freudenthal-Kuhn triangulations. An extensive discussion can be found in [8, Appendix A].

Both Coxeter and Freudenthal-Kuhn triangulations can be described as an arrangement of hyperplanes. They are related by an affine transformation. Let E be a finite set of vectors of $\mathbb{R}^d$ and consider the set of hyperplanes $H_E = \{x \in \mathbb{R}^d \mid \langle x, u\rangle = k, u \in E, k \in \mathbb{Z}\}$. Let, in addition, H be the hyperplane of $\mathbb{R}^{d+1}$ of equation $\langle x, \mathbf{1}\rangle = 0$ where $\mathbf{1}$ is the vector of $\mathbb{R}^{d+1}$ whose coordinates are all 1.

▶ **Definition 1.** *The Freudenthal-Kuhn triangulation is the hyperplane arrangement $\mathcal{H}_{E_{FK}}$ associated to the set of vectors $E_{FK} = \{e_1, \ldots, e_d\} \cup \{u_{i,j} = e_j - e_i \mid 1 \leq i < j \leq d\}$. The Coxeter triangulation of type $\tilde{A}_d$ is the hyperplane arrangement $\mathcal{H}_{E_C}$ in $\mathbb{R}^{d+1}$ associated to the set of vectors $E_C = \{r_{i,j} = e_i - e_{j+1} | \ 1 \leq i \leq j \leq d\}$, restricted to $H \simeq \mathbb{R}^d$.*

Two important facts follows. On one hand, because Coxeter and Freudenthal-Kuhn triangulations are related by an affine transformation, they have the same combinatorial structure. We call any triangulation that is the image of a Freudenthal-Kuhn triangulation under an affine transformation a CFK-triangulation. In this paper, we restrict our attention to Coxeter and Freudenthal triangulations since they are the simplest, but any CFK-triangulation could be used. The second fact is that each simplex in such a triangulation can be represented as a cell in an arrangement of $d(d-1)/2$ families of parallel hyperplanes which are known and do not need to be stored.

The next crucial observation relates CFK-triangulations and permutahedra, which allows to represent CFK-triangulations in a compact way (The definition and some combinatorial properties of permutahedra are given in [8, Appendix A]). We first recall that two complexes are dual if there is a bijection between their faces that inverses the inclusion relationships.

▶ **Proposition 2.** *The star of a vertex in a CFK-triangulation is combinatorially dual to a permutahedron.*

Since the facial structure of a permutahedron is fully described by ordered partitions, any simplex σ in a CFK-triangulation is characterized by a star that contains σ and by the ordered partition that specifies which simplex in the star is precisely σ. Since a simplex appears in several stars, we take the one that is centered at the lowest vertex of σ in the lexicographic order. This representation is called the permutahedral representation of a CFK-triangulation, see Figure 1. We further have:

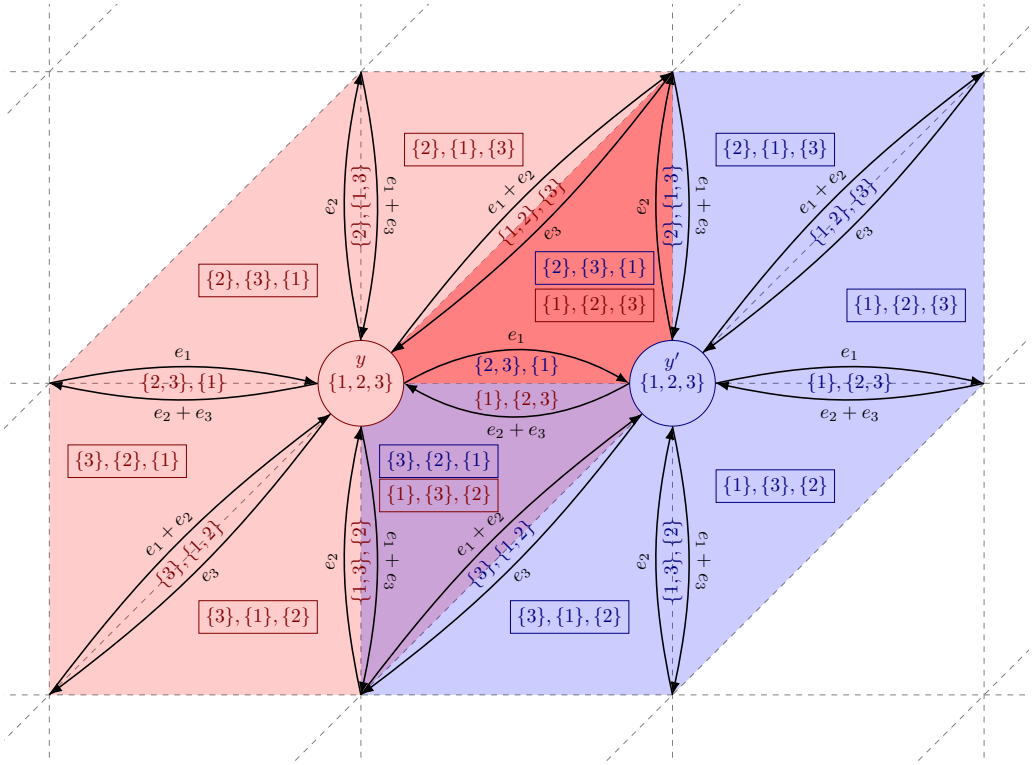


Figure 1 The permutahedral representation of the simplices in the stars of vertices y and y'.

▶ **Lemma 3** (Face computation). *Let σ be an l-simplex in the FK-triangulation of $\mathbb{R}^d$. Computing all its k-faces can be done in time $O(ds)$, where $s = \binom{l+1}{k+1}$ is the number of k-faces of an l simplex. The space complexity of the algorithm is $O(l)$.*

▶ **Lemma 4** (Coface computation). *Let σ be a k-simplex in the FK-triangulation of $\mathbb{R}^d$ given by its permutahedral representation. Computing the permutahedral representations of all its l-cofaces can be done in time $O(ds)$, where $s \leq \frac{1}{2^{\min(l,d-l)}} \binom{d-k}{d-l} (d-k+1)!$ is the number of l-cofaces of a k-simplex in the FK-triangulation. The space complexity of the algorithm is $O(d)$.*

2.2 PL-approximation of isomanifolds

We first recall sufficient conditions under which the PL-approximation $\hat{\mathcal{M}}$ output by the algorithm faithfully reproduces the original isomanifold. These conditions are fully described in the full version of [9] and we simply state here the main results specialized to the case of CFK-triangulations.

We will say that f has *bounded complexity* if the three following quantities $\gamma_{\max}$, $\lambda_{\min}$ and $\alpha_{\max}$ are positive and bounded.

$$\gamma_{\max} = \max_{x\in\mathcal{T}_0} (\max_i |\mathrm{grad} f^i(x)|) \quad \lambda_{\min} = \min_{x\in\mathcal{T}_0} \lambda_{\min}(x), \quad \alpha_{\max} = \max_{x\in\mathcal{T}_0} \max_i \|\mathrm{Hes}(f^i)(x)\|_2$$

where

- $\mathcal{T}_0$ denotes the set of all $\sigma \in \mathcal{T}$, such that $(f^i)^{-1}(0) \cap \sigma \neq \emptyset$ for all i.
- $\mathrm{grad} f^i = (\partial_j f_i)_j$ denotes the gradient of component f^i, for $i \in [1, d-n]$,
- $\mathrm{Gram}(\nabla f)$ denotes the Gram matrix whose elements are $\nabla f^i \cdot \nabla f^j$ where $\cdot$ stands for the dot product.
- $\lambda_{\min}(x)$ denotes the smallest absolute value of the eigenvalues of $\mathrm{Gram}(\nabla f(x))$,[1]
- $\mathrm{Hes}(f) = (\partial_k \partial_l f_i)_{k,l}$ denotes the Hessian matrix of second order derivatives,
- $|\cdot|$ denotes the Euclidean norm of a vector and $\|\cdot\|_2$ the operator 2-norm of a matrix.[2]

We can now restate the topological result of [9]:

▶ **Theorem 5.** *Assume that the function f has bounded complexity. If the precision of the CFK-triangulation satisfies $D = O(d^{-5/2})$, where the constant in the big O depends on $\gamma_{\max}$, $\lambda_{\min}$ and $\alpha_{\max}$, then $\hat{\mathcal{M}}$ is a manifold isotopic to the zero set $\mathcal{M}$ of f.*

Moreover, we can bound the Fréchet distance between $\mathcal{M}$ and $\hat{\mathcal{M}}$. The Fréchet distance is a quite strong notion of distance and, in particular, it bounds the Hausdorff distance.

▶ **Definition 6** (Fréchet distance for embedded manifolds)**.** *Let $\mathcal{M}_a$ and $\mathcal{M}_b$ be two homeomorphic, compact submanifolds of $\mathbb{R}^d$. Write $\mathcal{H}$ for the set of all homeomorphisms from $\mathcal{M}_a$ to $\mathcal{M}_b$. The Fréchet distance between $\mathcal{M}_a$ and $\mathcal{M}_b$ is $d_F(\mathcal{M}_a, \mathcal{M}_b) = \inf_{h\in\mathcal{H}} \sup_{x\in\mathcal{M}_a} d(x, h(x))$.*

▶ **Theorem 7.** *Assume that the function f has bounded complexity. Then, $d_F(\mathcal{M}, \hat{\mathcal{M}}) = O(D^2)$ where the constant in the big O depends on $\gamma_{\max}$, $\lambda_{\min}$ and $\alpha_{\max}$.*

3 Tracing isomanifolds

In this section, we describe an algorithm that computes a PL-approximation $\hat{\mathcal{M}}$ of an isomanifold $\mathcal{M}$. The algorithm has some similarity with the Marching Cube algorithm [20] but departs from it in two fundamental ways. First, because of the curse of dimensionality, we cannot afford to look at all the cells in the grid and need to limit the search to cells that are close to $\mathcal{M}$. The problem of computing $\hat{\mathcal{M}}$ can be naturally decomposed into two subproblems: locating the various components of $\mathcal{M}$ (i.e., finding at least one point in each connected component), and then tracing around each component, using the fact that the components are connected. This decomposition is used by various authors, see for example [27, 13]. In this paper, we focus on the tracing problem, although we discuss very briefly (Section 3.2) the problem of locating the components. As pointed out by Dobkin et al. many applications supply their own starting points.

[1] Because a Gram matrix is a symmetric square matrix, its eigenvalues are well defined and real.

[2] The operator norm is defined as $\|A\|_p = \max_{x\in\mathbb{R}^n} \frac{|Ax|_p}{|x|_p}$, with $|\cdot|_p$ the p-norm on $\mathbb{R}^n$.

The second major difference with the original marching cube algorithm is to replace the usual cubical grid by a CFK-triangulation of the ambient space. Taking a CFK-triangulation instead of a grid is a major advantage in high dimensions that has been recognized in the pioneering works of Allgower, and Schmidt [3] and of Dobkin et al. [13], see also [21]. The novelty here is to use the data structure of Section 2.1 to represent a CFK-triangulation. As a consequence, we will keep two main advantages of using grids: very limited storage and fast basic operations.

3.1 Isomanifolds

Let $f : \mathbb{R}^d \to \mathbb{R}^{d-n}$ be a smooth (C^2 suffices) function, and suppose that 0 is a regular value of f, meaning that at every point x such that $f(x) = 0$, the Jacobian of f is non-degenerate. Then the zero set of f is an n-dimensional manifold as a direct consequence of the implicit function theorem, see for example [14, Section 3.5]. We further assume that $f^{-1}(0)$ is compact. As in [1] we consider a triangulation $\mathcal{T}$ of $\mathbb{R}^d$. The function $\hat{f}$ is the linear interpolation of the values of f at the vertices if restricted to a single simplex $\sigma \in \mathcal{T}$, i.e.

$$\forall x \in \sigma : \quad \hat{f}(x) = \sum_{v \in \sigma} \lambda_v(x) f(v), \tag{1}$$

where the λ_v are the barycentric coordinates of x with respect to the vertices v of σ. For any function $g : \mathbb{R}^d \to \mathbb{R}^{d-n}$ we write g^i, with $i = 1, \ldots, d-n$, for the components of g.

The PL-approximation is now defined as $\hat{f}^{-1}(0) = \hat{\mathcal{M}}$. Locally, $\hat{f}|_\sigma^{-1}(0)$ is generically the intersection of an n-flat H_σ with σ. More precisely we note that $\hat{f}|_\sigma^{-1}(0)$ is an n-flat if the gradients of $\hat{f}^i|_\sigma$ are linearly independent, which can be easily achieved by perturbing f infinitesimally (or at least its values at the vertices). Let τ_j^{d-n} and τ_j^{d-n-1} be faces of σ of dimension $d-n$ and $d-n-1$. An infinitesimal perturbation of f, can prevent either $\hat{f}|_\sigma^{-1}(0)$ from intersecting the faces τ_j^{d-n-1}, or the gradients of $\hat{f}^i|_\sigma$ and the normal spaces of τ_j^{d-n} (for each fixed j) from failing to span $\mathbb{R}^d$. More precise statements on the geometric and topological stability of the triangulation under perturbations of f can be found in the full version of [9, Section 5]. Because $\hat{f}|_\sigma^{-1}(0)$ is (generically) the intersection of an n-flat (H_σ) and σ, it is an n-dimensional polytope denoted by C_σ. The PL-approximation or mesh $\hat{\mathcal{M}}$ of $\mathcal{M}$ is the polytopal cell complex obtained by gluing the polytopes C_σ associated to all the simplices σ in $\mathcal{T}$.

3.2 Manifold tracing algorithm

Let $\mathcal{M}$ be the zero set of some function $f : \mathbb{R}^d \to \mathbb{R}^{d-n}$, and let $\hat{\mathcal{M}}$ be the associated PL-approximation defined over a triangulation $\mathcal{T}$ of the ambient space $\mathbb{R}^d$. Both n, and d are known but arbitrary, and will be considered as parameters in the complexity analysis. We write $k = d - n$ for the codimension of $\mathcal{M}$. The algorithm will use for $\mathcal{T}$ a CFK-triangulation stored using the data structure from [8, Appendix A]. We assume that the manifold $\hat{\mathcal{M}}$, and the triangulation $\mathcal{T}$ satisfy the following genericity hypothesis:

▶ **Hypothesis 8** (Genericity). *Let σ be a d-simplex of $\mathcal{T}$ that intersects H_σ. No subface of σ of dimension less than k intersects H_σ, and any subface of σ of dimension k intersects H_σ in at most one point and transversally.*

We note that this condition can be satisfied by an infinitesimal perturbation for isomanifolds. This requires some explanation. We recall that the CFK-triangulation is a hyperplane arrangement, and up to translation there are a finite number of k-flats that contain all

k-simplices in the CFK triangulation. Hypothesis 8 is not satisfied, if either the flat H_σ is not linearly independent of these k-flats, or if H_σ does intersect some $(k-1)$-flat in the CFK-triangulation. In the previous section, we have already seen that an infinitesimal perturbation ensures that H_σ is n-dimensional. Because two affine spaces whose dimensions do not add up to the ambient dimension don't intersect with generically and two affine spaces whose dimensions add up to exactly the ambient dimension intersect in a single point, we see that genericity can be achieved by perturbing f infinitesimally. We further remark that, generically, any vertex of the PL-approximation $\hat{\mathcal{M}}$ is the intersection point between a k-simplex σ of $\mathcal{T}$ with the n-flat H_σ that interpolates f inside σ.

Algorithm 1 Manifold tracing algorithm.

input : the permutahedral representation of a triangulation $\mathcal{T}$ of $\mathbb{R}^d$,
the codimension of the isomanifold $k = d - n$,
a seed k-simplex τ_0 that intersects $\hat{\mathcal{M}}$

oracle : Given a k-simplex σ of $\mathcal{T}$, decide whether σ intersects H_σ and, in the affirmative, report the corresponding vertex $\sigma \cap H_\sigma = \sigma \cap \hat{\mathcal{M}}$.

output : Set $\mathcal{S}$ of the simplices in $\mathcal{T}$ of dimension k that intersect $\hat{\mathcal{M}}$, represented by their permutahedral representation, and the corresponding set $\hat{\mathcal{M}}_0$ of intersection points

```
1 Initialize the queue Q and the set S with τ_0
2 while the queue Q is not empty do
3     Pop a k-dimensional simplex τ from Q
4     foreach cofacet φ of τ do
5         foreach facet σ of φ do
6             if σ does not lie in S and intersects M̂ (which can be decided using the
               oracle) then
7                 Insert σ into the queue Q
8                 Insert σ into S together with the intersection point provided by the
                   oracle
```

The algorithm essentially computes the set $\mathcal{S}$ of k-simplices of $\mathcal{T}$ that intersect $\hat{\mathcal{M}}$. The elements of $\mathcal{S}$ are in 1-1 correspondence with the vertices of $\hat{\mathcal{M}}$ thanks to the Genericity hypothesis. The so-called intersection oracle is a basic ingredient of the algorithm:

Intersection oracle: *Given a k-simplex σ of $\mathcal{T}$, decide whether σ intersects H_σ and, in the affirmative, report the corresponding vertex $\sigma \cap H_\sigma$.*

It is easy to see that the intersection oracle reduces to solving a linear system. Indeed, generically, a vertex is the intersection of a k-simplex σ of $\mathcal{T}$ with the m-flat H_σ that interpolates f inside σ. One can compute the barycentric coordinates of $\sigma \cap H_\sigma$ by solving a linear system of k equations, and k unknowns. It then remains to check whether the barycentric coordinates are all non-negative (to ensure that the intersection point lies inside σ). It follows that the intersection oracle reduces to evaluating f at the $k+1$ vertices of σ plus solving a $k \times k$ linear system.

In addition, we need to provide a set of k-simplices of $\mathcal{T}$ to initialize the tracing. These simplices must intersect all the connected components of the isomanifold and are called seed simplices. If $\mathcal{M}$ consists of multiple connected components, then a seed simplex must be provided per each connected component and we proceed in the same manner for each component. So we will assume for now that $\mathcal{M}$ is connected.

The seed simplices are given as part of the input and we don't discuss in this paper the problem of their construction. We simply observe that they can be obtained by computing a critical point (e.g., a point with smallest x_1-coordinate) on each connected component of the isomanifold, which reduces to finding a solution to a system of equations, on which a large body of literature exists. See for example [24, 23, 13] and also the discussion in Wenger's book [27, Section 8.4]. Once such a seed point has been computed, we simply translate and rotate the triangulation $\mathcal{T}$ so that the seed point coincides with the barycenter of a k-simplex of $\mathcal{T}$ and the intersection with the manifold is transversal as demanded by the genericity hypothesis (for numerical stability it is convenient if the angle between the tangent space of the manifold and the starting k-simplex is large, which is easy to ensure). If the distance between $\mathcal{M}$ and $\hat{\mathcal{M}}$ is small enough, then $\hat{\mathcal{M}}$ also intersects the same k simplex, see Section 2.2.

The algorithm is described as Algorithm 1. It takes as input the permutahedral representation of an ambient FKC-triangulation $\mathcal{T}$ and a seed k-simplex τ_0 of $\mathcal{T}$. We assume that $\mathcal{T}$ satisfies the Genericity Hypothesis 8, which can be enforced by infinitesimal perturbations of f as discussed in Section 3.1.

The algorithm maintains the subset $\mathcal{S}$ of the simplices in $\mathcal{T}$ of dimension k that intersect $\hat{\mathcal{M}}$. $\mathcal{S}$ is initialized with the seed simplex τ_0 and stored as a hash table so that we can decide in constant time if a given k-simplex belongs to $\mathcal{S}$. Then, starting from τ_0, we look at all its cofacets and consider all the facets of those cofacets that are not in $\mathcal{S}$ (i.e. they have not been considered yet). This can be done using a queue $\mathcal{Q}$ of candidate k-simplices. Each of these simplices is queried with the intersection oracle and, if it is found to intersect $\hat{\mathcal{M}}$, it is added to $\mathcal{S}$ if not already present. Upon termination, $\mathcal{S}$ contains all the k-dimensional simplices of $\mathcal{T}$ that intersect $\hat{\mathcal{M}}$. Each such intersection, which consists of a single point (by the Genericity hypothesis), is a vertex of $\hat{\mathcal{M}}$. Hence $\hat{\mathcal{M}}_0$ is the vertex set of $\hat{\mathcal{M}}$.

Note that our algorithm essentially traverses the adjacency graph of the k and $(k+1)$-simplices of $\mathcal{T}$ that intersect $\hat{\mathcal{M}}$. It therefore identifies not only the set $\hat{\mathcal{M}}_0$ of vertices of $\hat{\mathcal{M}}$, but also the edges joining two such vertices (associated to the cofacets of the k-simplices in $\mathcal{S}$). By simply reporting those cofacets on the fly, the algorithm can output the 1-skeleton $\hat{\mathcal{M}}_1$ of the n-dimensional polytopal cell complex $\hat{\mathcal{M}}$. The higher dimensional faces of $\hat{\mathcal{M}}$ are the polytopes $C_\tau = \tau \cap H_\tau$ for all the cofaces τ of the k-simplices of $\mathcal{S}$. If needed, the full Hasse diagram of $\hat{\mathcal{M}}$ can be computed from $\hat{\mathcal{M}}_0$. This can be done in an output sensitive manner by using the permutahedral representation of $\mathcal{T}$ and the algorithm of [8, Appendix A] to compute cofaces by increasing dimensions.

3.3 Complexity analysis

We can easily bound the complexity of the manifold tracing algorithm as a function of the size of the output.

▶ **Proposition 9.** *The time complexity of the algorithm is $O(k2^n I|\mathcal{S}|)$ where I is the time complexity of one call of the intersection oracle, and $|\mathcal{S}|$ is the number of simplices of dimension k output by the algorithm.*

Since, the intersection oracle reduces to evaluating f at the $k+1$ vertices of σ plus solving a $k \times k$ linear system, $I = O(k^\omega)$ where $\omega \approx 2.375$.

We will now express the size of the output in terms of quantities that depend on the manifold, the ambient dimension d, and the resolution of the triangulation (the diameter D of a simplex) which bounds the density of the output sample, and the precision of the

approximation. Our result holds for K-sparse manifolds, i.e. submanifolds whose intersection with any k-flat consists of at most K points. In practical situations, K is usually small and, in particular, K is a constant for algebraic isomanifolds of bounded degree.

▶ **Proposition 10** (Size of the output). *Assume that $\mathcal{M}$ is contained in the unit cube $C_d = [0,1]^d$, and that any k-flat intersects $\mathcal{M}$ at most K times. Writing $|\mathcal{S}| = N_C$ when $\mathcal{T}$ is a Coxeter triangulation and $|\mathcal{S}| = N_{FK}$ for a Freudenthal triangulation, we have $N_C \leq \frac{K}{n!} \times \left(\frac{d^2\sqrt{d(d+2)}}{2\sqrt{2}D}\right)^n$ and $N_{FK} \leq \frac{K}{n!} \times \left(\frac{d^3}{\sqrt{2}D}\right)^n$ where D is the diameter of a simplex of $\mathcal{T}$.*

We see that Coxeter triangulations lead to smaller samples than FK-triangulations by a factor of roughly 2^n. This will be confirmed experimentally (see Figure 4).

As noticed in Section 3.2, a simple variant of the algorithm can compute the full Hasse diagram of $\hat{\mathcal{M}}$ in an output sensitive manner. The following lemma shows that the combinatorial complexity of $\hat{\mathcal{M}}$ is of the same order as the combinatorial complexity as $\hat{\mathcal{M}}_0$.

▶ **Proposition 11.** *The combinatorial complexity of $\hat{\mathcal{M}}$ is $|\mathcal{S}| \times (\frac{3}{2})^n (n+1)!$, where $|\mathcal{S}|$ is bounded in Proposition 10. If $n = O(1)$, the combinatorial complexity of $\hat{\mathcal{M}}$ is polynomial in d, and $\delta = 1/D$.*

We combine Propositions 9, 10, and 11 to obtain our main result.

▶ **Theorem 12.** *Assume that $\mathcal{M}$ is contained in the unit cube $[0,1]^d$ and that any affine k-flat intersects $\mathcal{M}$ at most K times (K is usually small, and is in particular a constant for algebraic isomanifolds of bounded degree). Let, in addition, D be the precision required on the approximation (the diameter of a simplex in the ambient triangulation $\mathcal{T}$). The size of the output, and the time complexity of the algorithm are polynomial in the ambient dimension d, and in $\delta = 1/D$, and exponential in the intrinsic dimension n. The same result holds for the full PL-approximation $\hat{\mathcal{M}}$ of $\mathcal{M}$.*

3.4 Dimensionality reduction

As seen from Proposition 10, the size $|\mathcal{S}|$ of the output of the algorithm, considered as a function of the resolution D of the triangulation, depends exponentially on n (which is to be expected), and only polynomially on d (which is fortunate). Nevertheless, the computing time of our algorithm and the size of the output depend on d. Removing the dependency on d in the time complexity is impossible since we need to evaluate a vector-valued function f at a number of points of $\mathbb{R}^d$, which takes $\Omega(d)$ time per evaluation. However, we will see that we can reduce the size of the mesh produced by our algorithm.

Examples of samples of $\mathcal{M}$ whose sizes depend on n but not on d, and lead to good approximations are known. Especially important are D-nets [11, 6]. A D-net consists of a finite number of sample points of $\mathcal{M}$ such that no point of $\mathcal{M}$ is at distance more than D from a sample point (density condition), and no two sample points are closer than cD for some positive constant c (separation condition). A simple volume argument shows that the size of a D-net of a n-dimensional smooth submanifolds is $O(1/D^n)$ [5, Lemma 5.3]. The sample produced by our algorithm is D-dense on the piecewise linear approximation. This implies that we have a sample that has a Hausdorff distance of $D + d_F(\mathcal{M}, \hat{\mathcal{M}})$ to the manifold, where $d_F(\mathcal{M}, \hat{\mathcal{M}})$ is bounded in Theorem 7.

Since its cardinality depends on d, it is not well separated and, in particular, not a D-net of $\mathcal{M}$. If we are mostly interested in the output sample, we can easily sparsify it to obtain a D-net. However, by doing so, we will lose the combinatorial structure of the mesh.

We now show how to compute a D-dense sample of $\mathcal{M}$ of size independent of d, together with a mesh. Specifically, we will reduce dimensionality using a variant of the celebrated Johnson-Lindenstrauss lemma for manifolds. Doing so, we depart from our previous worst-case analysis by allowing some approximation factor ε and tolerate a guarantee that holds only with high probability.

▶ **Theorem 13** (Johnson-Lindenstrauss lemma for manifolds [12, 26]). *Pick any $\varepsilon, \eta > 0$, and let $d' = \Omega\left(\frac{n}{\varepsilon^2}\log\frac{1}{\varepsilon} + \frac{1}{\varepsilon^2}\log\frac{\Gamma}{\eta}\right)$, where Γ is a quantity that depends only on intrinsic properties of $\mathcal{M}$. Let Φ be the projection on a random affine subspace of dimension d'. Then, with probability $> 1-\eta$, for all $x, y \in \mathcal{M}$, we have $(1-\varepsilon)\sqrt{\frac{d'}{d}} \leq \frac{\|\Phi x - \Phi y\|}{\|x-y\|} \leq (1+\varepsilon)\sqrt{\frac{d'}{d}}$.*

Let $\Psi = \sqrt{\frac{d}{d'}}\,\Phi$. By the theorem, the image $\Psi(\mathcal{M})$ of $\mathcal{M}$ is a submanifold of dimension n embedded in $\mathbb{R}^{d'}$. One can now run the manifold tracing algorithm in $\mathbb{R}^{d'}$ to sample, and mesh $\Psi(\mathcal{M})$. The algorithm works as described before except that we need another oracle that, given a $(d'-n)$-simplex σ of the CFK-triangulation of $\mathbb{R}^{d'}$, decides whether its inverse image $\Psi^{-1}(\sigma)$ intersects $\mathcal{M}$ or not. Note that $\Psi^{-1}(\sigma)$ is a $(d-d')$-dimensional flat strip (that is the product of a face and an affine subspace) in $\mathbb{R}^d$, and that the complexity of this new oracle is the same as the complexity of the basic intersection oracle, i.e. polynomial in d.

Due to the scaling factor $\sqrt{d/d'}$, the resolution of the triangulation in the low dimensional space $\mathbb{R}^{d'}$ has to be scaled by the same factor if one wants to satisfy a given sampling density on $\mathcal{M}$. Since the geometry of the manifold is also scaled in the same way [16], the analysis of the algorithm will be unchanged. Proposition 10 then shows that the size of the output sample does not depend on d but only on n and D for fixed ε, and η. Moreover, since the complexities of the projection and of the new oracle are polynomial in d, Proposition 9 implies that the overall complexity is still polynomial in d.

3.5 Isomanifolds with boundary, and isostratifolds

The case of isomanifolds with boundary and, more generally, of isostratifolds can be handled in very much the same way. By an isomanifold of dimension n with boundary, we mean that, on top of a function $f : \mathbb{R}^d \to \mathbb{R}^{d-n}$, we are given another function $f_\partial : \mathbb{R}^d \to \mathbb{R}$, and the set we consider is $\mathcal{M} = f^{-1}(0) \cap f_\partial^{-1}([0,\infty))$. We note that $\partial\mathcal{M} = f^{-1}(0) \cap f_\partial^{-1}(0)$.

Similarly to (1), we also define $\hat{f}_\partial|_\tau(x) = \sum_{v\in\sigma} \lambda_v(x)\, f_\partial(v)$. We write $\hat{f}$ for the (global) piecewise linear function that coincides with $\hat{f}|_\tau$ on each τ of $\mathcal{T}$, and $\hat{f}_\partial$ for the (global) piecewise linear function that coincides with $\hat{f}_\partial|_\tau$ on each τ of $\mathcal{T}$. We note that the piecewise linear approximation of the boundary $\partial\hat{\mathcal{M}} = \hat{f}_\partial^{-1}(0) \cap \hat{f}^{-1}(0)$ is a subset of $\hat{f}^{-1}(0)$, i.e. the piecewise linear approximation of the manifold ignoring the boundary. The piecewise linear approximation $\hat{\mathcal{M}}$ of the manifold with boundary consists of the following cells:

- For each τ of $\mathcal{T}$, such that $\hat{f}_\partial|_\tau$ is positive on τ, and $(\hat{f}|_\tau)^{-1}(0) \cap \tau \neq \emptyset$, we add $(\hat{f}|_\tau)^{-1}(0) \cap \tau$.
- For each τ of $\mathcal{T}$, such that $(\hat{f}|_\tau)^{-1}(0) \cap \tau \neq \emptyset$, and $(\hat{f}_\partial|_\tau)^{-1}(0) \cap \tau \neq \emptyset$, we add $(\hat{f}|_\tau)^{-1}(0) \cap (\hat{f}_\partial|_\tau)^{-1}([0,\infty)) \cap \tau$.

We will assume that the Genericity Hypothesis 8 holds for both $\hat{\mathcal{M}}$, and $\partial\hat{\mathcal{M}}$.

We can now adapt the algorithm of Section 3.2 as follows. In addition to reporting the set S_k of k-faces of the triangulation $\mathcal{T}$ that intersect $\hat{\mathcal{M}}$, the algorithm will also report the set S_{k+1} of $(k+1)$-faces of the triangulation $\mathcal{T}$ that intersect $\partial\hat{\mathcal{M}}$. The computation of S_{k+1} is done by the following simple modification of Algorithm 1: if the k-dimensional facet σ of τ intersects $\hat{f}^{-1}(0)$ at a point x such that $\hat{f}_\partial|_\tau(x) < 0$ (i.e. x is not in $\hat{\mathcal{M}}$), we then compute the intersection point of τ with $\hat{f}_\partial^{-1}(0)$, and put τ in S_{k+1}.

As for the case of manifolds without boundary (see the discussion at the end of Section 3.2), the algorithm traverses (and therefore computes) the 1-skeleton of $\hat{\mathcal{M}}$. Under the Genericity Hypothesis 8, the vertices of $\hat{\mathcal{M}}_1$ are in bijection with the simplices of $S_k \cup S_{k+1}$. The edges are obtained by applying the following rules below (we identify a simplex in S_k (resp. S_{k+1}) and the intersection point $S_k \cap \hat{\mathcal{M}}$ (resp. $S_{k+1} \cap \partial\hat{\mathcal{M}}$):

1. Two simplices σ_1, and σ_2 of S_k are joined by an edge in $\hat{\mathcal{M}}_1$ if and only if there exists a simplex in $\mathcal{T}_{k+1}$ with faces σ_1 and σ_2.
2. Two simplices τ_1, and τ_2 of S_{k+1} are joined by an edge in $\widehat{\partial\mathcal{M}}_1$ if and only if there exists a simplex in $\mathcal{T}_{k+2}$ with faces τ_1 and τ_2.
3. A simplex σ of S_k, and a simplex τ of S_{k+1} are joined by an edge in $\widehat{\partial\mathcal{M}}_1$ if and only if σ is a facet of τ.

The three rules above together with the permutahedral representation of $\mathcal{T}$ provide a way to construct the 1-skeleton of $\hat{\mathcal{M}}$ on the fly. The total cost is output sensitive. If needed, the entire combinatorial structure of $\hat{\mathcal{M}}$ can be computed by traversing the full triangulation $\mathcal{T}$.

The above construction generalizes easily to arbitrary isostratifolds. Isostratifolds are stratified spaces that are defined by equations and inequalities. An example of such an isostratifold is an octant of the sphere in $\mathbb{R}^3$ that can be defined by as $x^2 + y^2 + z^2 - 1 = 0$, $x \geq 0$, $y \geq 0$, and $z \geq 0$. We compute the 1-skeleton of $\hat{\mathcal{M}}$ and construct a graph whose nodes are the simplices of dimensions $k, k+1, ..., d$ that intersect the strata of dimension $n, n-1, ..., 0$.

4 Experimental results

The algorithm of Section 3 has been implemented in C++. The code is robust and fast and will be released in the GUDHI library [18]. Full detail on the implementation, including the implementation of the oracle, can be found in [19].

In this section, we explore the dependency of our C++ implementation of the data structure for the ambient CFK-triangulation, and of the manifold tracing algorithm on the properties of the triangulation, and of the input manifold.

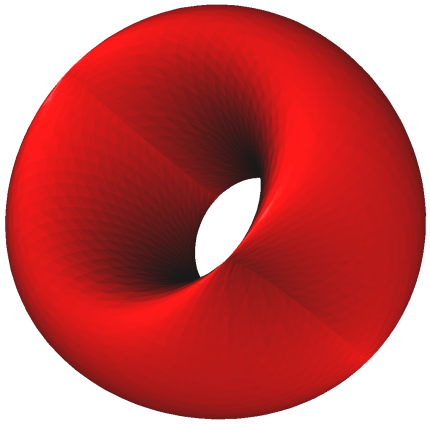


Figure 2 The piecewise-linear approximation of a flat torus embedded in $\mathbb{R}^{10}$ defined by the equations $x_1^2 + x_2^2 = 1$, and $x_3^2 + x_4^2 = 1$, and $x_i = 0$ for $i > 4$, projected to $\mathbb{R}^3$. The ambient triangulation used is a Coxeter triangulation of type $\tilde{A}_{10}$ with the diameter of the full-dimensional simplices 0.23. The output size $|\mathcal{S}|$ is 509 952. The execution time of the algorithm is 231s. The torus has been rotated and translated in $\mathbb{R}^{10}$ so that the coordinate axes do not play any special role.

4.1 Performance of the algorithm

We show the performance of our implementation of the manifold tracing algorithm for various ambient and intrinsic dimensions in Figure 3. In Figure 4, we can see that using Coxeter triangulation is beneficial in practice as it produces a smaller output in less time (see Proposition 10).

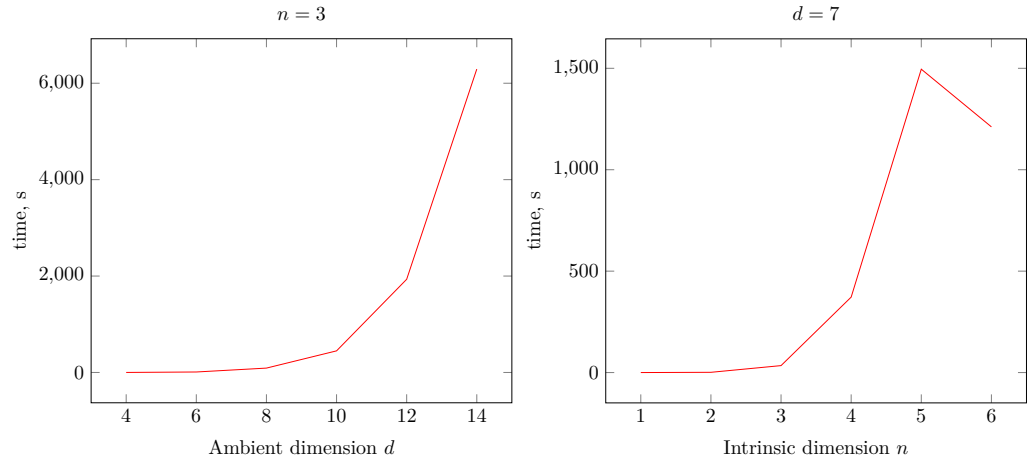


Figure 3 The effect of the ambient dimension d and of the intrinsic dimension n on the computation time of the manifold tracing algorithm. The reconstructed manifold in the tests is the n-dimensional sphere embedded in $\mathbb{R}^d$. The ambient triangulation used is a Coxeter triangulation of type $\tilde{A}_d$. The diameter of the full simplices is fixed for all d.

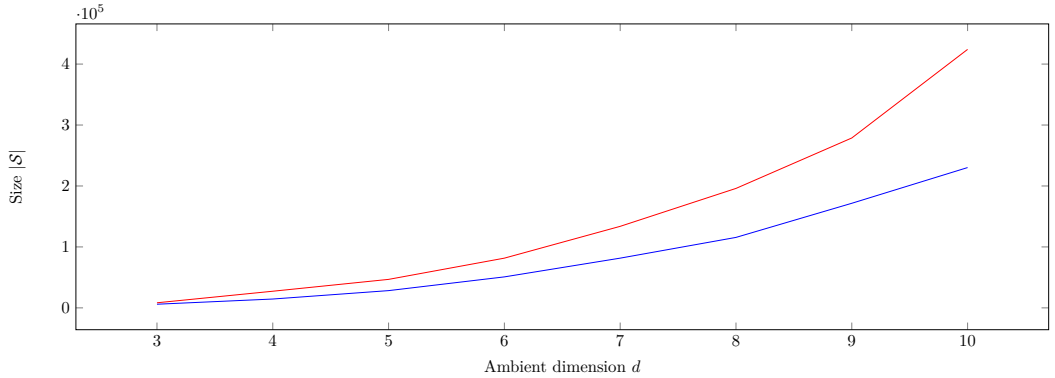


Figure 4 Comparison of the size of the output of the manifold tracing algorithm using two types of ambient triangulations: a Coxeter triangulation of type $\tilde{A}_d$ (in blue), and the Freudenthal-Kuhn triangulation of $\mathbb{R}^d$ (in red) with the same diameter $0.07\sqrt{d}$ of d-dimensional simplices. The reconstructed manifold is the 2-dimensional implicit surface "Chair" embedded in $\mathbb{R}^d$ given by the equations: $(x_1^2 + x_2^2 + x_3^2 - 0.8)^2 - 0.4\left((x_3 - 1)^2 - 2x_1^2\right)\left((x_3 + 1)^2 - 2x_2^2\right) = 0$, and $x_i = 0$ for $i > 3$.

In Figure 2, we present a PL approximation of a two-dimensional Clifford torus without boundary embedded in $\mathbb{R}^{10}$ built by the manifold tracing algorithm. The torus has been rotated and translated in $\mathbb{R}^{10}$ so that the coordinate axes do not play any special role. Note that there is no C^2 isometric embedding of the Clifford torus in $\mathbb{R}^3$.

4.2 Manifolds with boundary

The algorithm has been adapted to handle submanifolds with boundary and surfaces with a piecewise smooth boundary, see Section 3.5. In Figure 5, we present the mesh obtained by our algorithm on a portion of a flat torus embedded in $\mathbb{R}^4$, and cut by a hypersphere. The torus has been rotated and translated in $\mathbb{R}^4$ so that the coordinate axes do not play any special role.

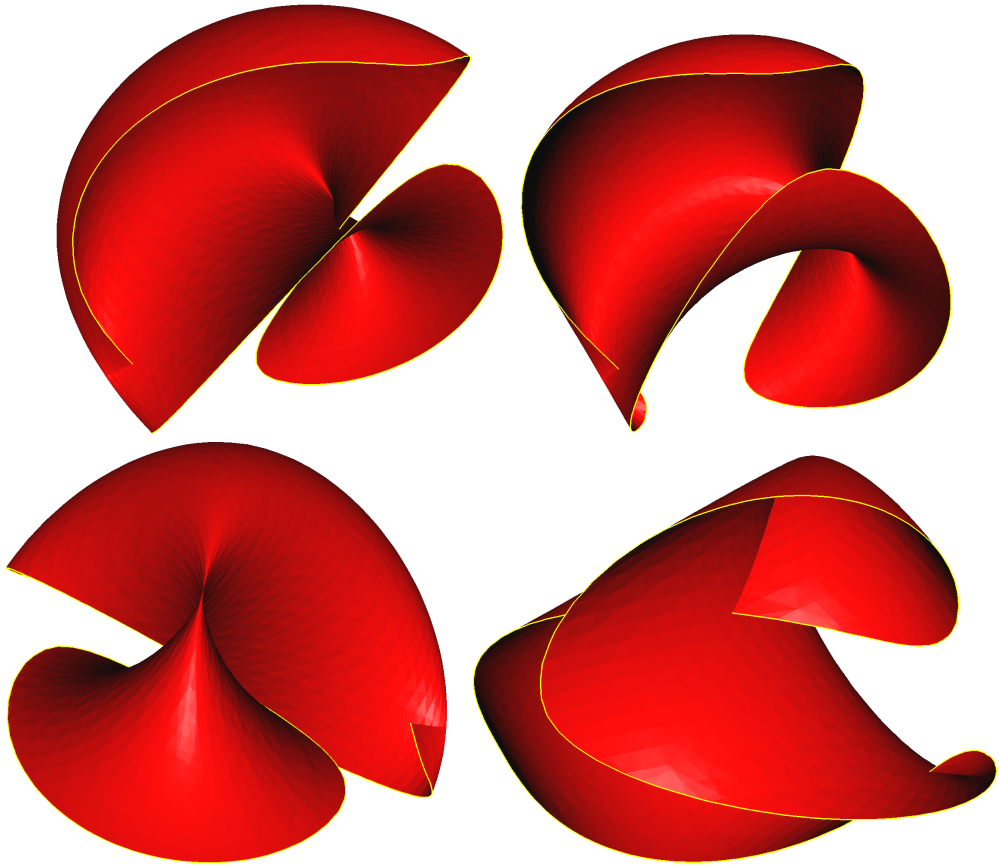


Figure 5 Four views of the flat torus in $\mathbb{R}^4$ given by two equations $x_1^2+x_2^2=1$, and $x_3^2+x_4^2=1$ cut by the hypersphere $(x_1-1)^2+x_2^2+(x_3-1)^2+x_4^2=4$, projected to $\mathbb{R}^3$. The ambient triangulation used is a Coxeter triangulation of type $\tilde{A}_4$ with the diameter 0.15 of the full-dimensional simplices. The reconstructed boundary is highlighted in yellow. The size $|\mathcal{S}|$ of the piecewise-linear approximation is 14 779. The execution time of the algorithm is 1.84s. The torus has been rotated and translated in $\mathbb{R}^4$ so that the coordinate axes do not play any special role.

4.3 An application in algebraic geometry

We also applied our algorithm to a more complicated example of interest in algebraic geometry [4] where an active field of research is to understand the geometry and topology of various projective varieties. Projective varieties are isomanifolds defined by polynomial equations in the complex projective space $\mathbb{CP}^d = (\mathbb{C}^{d+1} \setminus 0)/\mathbb{C}^*$ of complex dimension d. One such example is the complex one-dimensional curve (that is a real dimensional surface) given by the equation $z_1^2\bar{z}_2 + z_2^2\bar{z}_3 + z_3^2\bar{z}_1 = 0$ in $\mathbb{CP}^2$, where $\bar{z}$ denotes the conjugate of the complex number z.

To be able to apply our algorithm, we first need to pass from homogenous coordinates $[z_1 : \ldots : z_{d+1}]$ on $\mathbb{CP}^d$ to affine coordinates $[z'_1 : \cdots : z'_{i-1} : 1 : z'_{i+1} : \cdots : z'_{d+1}]$ by picking the ith coordinate to be equal to 1, that is $z'_j = z_j/z_i$. Given some homogenous coordinates $[z_1 : \ldots : z_{d+1}]$, we can choose the ith coordinate to be set to 1 to be the coordinate whose absolute value is the largest, so that $\mathbb{CP}^d$ can be written as the union of the $d+1$ sets $\left\{[z'_1 : \cdots : z'_{i-1} : 1 : z'_{i+1} : \cdots : z'_{d+1}] \mid |z'_j| \leq 1\right\}$, with the boundaries of these sets identified. Writing $z'_j = x_j + iy_j$ these sets are (seen as real sets) identical to the domain of $\mathbb{R}^{2d}$

$$D_i = \{(x_1, y_1, \ldots, x_{i-1}, y_{i-1}, x_{i+1}, y_{i+1}, \ldots, x_{d+1}, y_{d+1}) \mid x_j^2 + y_j^2 \leq 1\}.$$

Let f be a homogenous polynomial in $d+1$ complex variables and their complex conjugates. For each i, we can fix the ith coordinate to be 1. Writing each variable in terms of its real and imaginary part yields a real inhomogeneous polynomial in $2d$ (real) variables on the domain D_i. Taking the real and imaginary parts of the function yields two real functions $f_{\Re,i}$ and $f_{\Im,i}$ on D_i. As real sets, the projective variety $f = 0$ on $\mathbb{CP}^d$ and the intersection of the sets $f_{\Re,i} = 0$ and $f_{\Im,i} = 0$ on D_i for each i (with the boundaries identified) are the same. We can therefore apply the tracing algorithm to each isomanifold $(f_{\Re,i} = 0, f_{\Im,i} = 0)$ of D_i independently. Since their boundaries coincide, we can then glue these isomanifolds along their boundary to obtain a PL-approximation of the projective variety $f = 0$. This, for example, allows to recover the Euler characteristic of $f = 0$ on $\mathbb{CP}^d$.

This principle generalizes to varieties of higher codimension, that is to varieties defined by a number of homogenous polynomials $f_1, \ldots, f_{d-m}$.

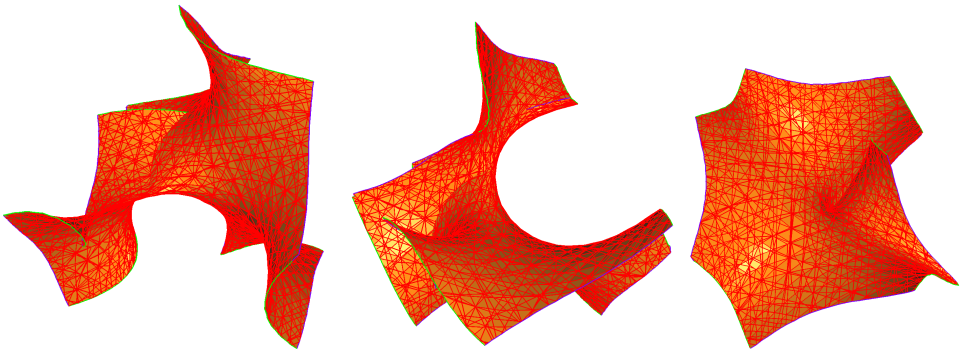


Figure 6 The three triangulated surfaces as discussed in the example of $z_1^2\bar{z}_2 + z_2^2\bar{z}_3 + z_3^2\bar{z}_1 = 0$ in $\mathbb{CP}^2$ after projection from $\mathbb{R}^4$ to $\mathbb{R}^3$.

We illustrate the above construction on the above equation $z_1^2\bar{z}_2 + z_2^2\bar{z}_3 + z_3^2\bar{z}_1 = 0$ in $\mathbb{CP}^2$. By passing to affine coordinates, we recover $z_1^2\bar{z}_2 + z_2^2 + \bar{z}_1 = 0$, $z_1^2 + \bar{z}_3 + z_3^2\bar{z}_1 = 0$, and $\bar{z}_2 + z_2^2\bar{z}_3 + z_3^2 = 0$. By expanding $z_1 = x_1 + iy_1$, $z_2 = x_2 + iy_2$, and $z_3 = x_3 + iy_3$, we find two real equations for each of the complex equations. We give those corresponding to $z_1^2\bar{z}_2 + z_2^2 + \bar{z}_1 = 0$, the other equations being symmetric. For this complex equation, we get the real equations $x_1 + x_1^2x_2 + x_2^2 - x_2y_1^2 + 2x_1y_1y_2 - y_2^2 = 0$ and $-y_1 + 2x_1x_2y_1 - x_1^2y_2 + 2x_2y_2 + y_1^2y_2 = 0$ in $\mathbb{R}^4$. The domain D_3 is in this case determined by the equations $x_1^2 + y_1^2 \leq 1$ and $x_2^2 + y_2^2 \leq 1$. Hence we find a surface in $\mathbb{R}^4$ with a piecewise smooth boundary. The result provided by our algorithm is shown in Figure 6. For visualization purposes, we show the three surfaces separately and projected from $\mathbb{R}^4$ to $\mathbb{R}^3$.

5 Conclusion and open questions

We have presented an efficient, practical and provably correct algorithm to compute the PL-approximation of an isomanifold of any dimension and codimension. Since isomanifolds are a special type of manifolds, it is tempting to see if our algorithm extends to general smooth submanifolds of $\mathbb{R}^d$.

The manifold tracing algorithm itself is quite general and works for any submanifold as soon as we provide a seed point and an oracle that can determine whether a k-simplex of the ambient triangulation intersects $\mathcal{M}$ or not. In this general setting, the simple algorithm described above is sufficient to compute a PL-approximation of the manifold and satisfies the bounds given in Section 3.

However, this is not enough to obtain guarantees on the geometric and topological quality of the output mesh. Such guarantees can be obtained by slightly perturbing the ambient Coxeter triangulation of type $\tilde{A}_d$ so that the following conditions are satisfied:

1. All k-dimensional faces τ in $\mathcal{T}$, with $k \leq d-n-1$, are far enough from $\mathcal{M}$.
2. The longest edge length of $\mathcal{T}$ is upper bounded and its smallest thickness is lower bounded.

Under these conditions, Algorithm 1 will output a PL-approximation that is topologically equivalent and close in Hausdorff distance to the input manifold [7]. However, the perturbation scheme of [7] perturbs (in the worst case) all the simplices of $\mathcal{T}$ of dimension less than the codimension $d-n$ that are incident on a vertex (in a neighbourhood of $\mathcal{M}$). Since there are exponentially many such simplices, such methods have a complexity that depends exponentially on the ambient dimension d, and have not proved useful in practice except in some simple cases. It remains open whether general smooth manifolds embedded in $\mathbb{R}^d$ can be triangulated in time polynomial in d as we were able to do here in the special case of isomanifolds.

References

1 Eugene Allgower and Kurt Georg. Estimates for piecewise linear approximations of implicitly defined manifolds. *Applied Mathematics Letters*, 2(2):111–115, 1989. doi:10.1016/0893-9659(89)90001-3.

2 Eugene Allgower and Kurt Georg. *Numerical continuation methods: an introduction*, volume 13. Springer Science & Business Media, 1990. doi:10.1007/978-3-642-61257-2.

3 Eugene Allgower and Phillip H. Schmidt. An algorithm for piecewise-linear approximation of an implicitly defined manifold. *SIAM Journal on Numerical Analysis*, 22(2):322–346, 1985. doi:10.1137/0722020.

4 Aurélien Alvarez and Bertrand Deroin. Dynamique et topologie du feuilletage de Jouanolou. Preprint, 2019.

5 Jean-Daniel Boissonnat, Frédéric Chazal, and Mariette Yvinec. *Geometric and Topological Inference*. Cambridge Texts in Applied Mathematics. Cambridge University Press, 2018. doi:10.1017/9781108297806.

6 Jean-Daniel Boissonnat and Arijit Ghosh. Manifold reconstruction using tangential Delaunay complexes. *Discrete & Computational Geometry*, 51(1):221–267, 2014. doi:10.1007/s00454-013-9557-2.

7 Jean-Daniel Boissonnat, Siargey Kachanovich, and Mathijs Wintraecken. Triangulating submanifolds: An elementary and quantified version of whitney's method. *Discrete & Computational Geometry*, pages 1–49, 2020. doi:10.1007/s00454-020-00250-8.

8 Jean-Daniel Boissonnat, Siargey Kachanovich, and Mathijs Wintraecken. Tracing Isomanifolds in $\mathbb{R}^d$ in Time Polynomial in d using Coxeter-Freudenthal-Kuhn Triangulations. Full version of this paper, 2021. URL: https://hal.inria.fr/hal-03006663.

9 Jean-Daniel Boissonnat and Mathijs Wintraecken. The topological correctness of PL-approximations of isomanifolds. In *34th International Symposium on Computational Geometry, SoCG 2020, June 23-26, 2020, Zurich, Switzerland.*, 2020. Full version. URL: https://hal.inria.fr/hal-02386193.

10 Yen-Chi Chen. Solution manifold and its statistical applications, 2020. arXiv:2002.05297. arXiv:2002.05297.

11 Siu-Wing Cheng, Tamal K Dey, and Edgar A Ramos. Manifold reconstruction from point samples. In *SODA*, pages 1018–1027, 2005.

12 Kenneth L. Clarkson. Tighter bounds for random projections of manifolds. In *Proceedings of the 24th ACM Symposium on Computational Geometry, College Park, MD, USA, June 9-11, 2008*, pages 39–48, 2008. doi:10.1145/1377676.1377685.

13 David P. Dobkin, Allan R. Wilks, Silvio V. F. Levy, and William P. Thurston. Contour tracing by piecewise linear approximations. *ACM Transactions on Graphics (TOG)*, 9(4):389–423, 1990. doi:10.1145/88560.88575.

14 J. J. Duistermaat and J. A. C. Kolk. *Multidimensional Real Analysis I: Differentiation*. Number 86 in Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2004. doi:10.1017/CBO9780511616716.

15 B. Curtis Eaves. *A course in triangulations for solving equations with deformations*, volume 234. Lecture Notes in Economics and Mathematical Systems, 1984. doi:10.1007/978-3-642-46516-1.

16 Armin Eftekhari and Michael B. Wakin. What happens to a manifold under a bi-Lipschitz map? *Discrete & Computational Geometry*, 57(3):641–673, 2017. doi:10.1007/s00454-016-9847-6.

17 A. Gomes, I. Voiculescu, J. Jorge, B. Wyvill, and C. Galbraith. *Implicit Curves and Surfaces: Mathematics, Data Structures and Algorithms*. Springer, 2009. doi:10.1007/978-1-84882-406-5.

18 GUDHI Project. URL: http://gudhi.gforge.inria.fr/doc/latest/.

19 Siargey Kachanovich. *Meshing submanifolds using Coxeter triangulations*. Theses, COMUE Université Côte d'Azur (2015 - 2019), 2019. URL: https://hal.inria.fr/tel-02419148.

20 William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169, 1987. doi:10.1145/37401.37422.

21 Chohong Min. Simplicial isosurfacing in arbitrary dimension and codimension. *Journal of Computational Physics*, 190(1):295–310, 2003. doi:10.1016/S0021-9991(03)00275-4.

22 Timothy S. Newman and Hong Yi. A survey of the marching cubes algorithm. *Computers & Graphics*, 30(5):854–879, 2006. doi:10.1016/j.cag.2006.07.021.

23 James M Ortega and Werner C Rheinboldt. *Iterative solution of nonlinear equations in several variables*. Classics in Applied Mathematics. SIAM, 2000. doi:10.1137/1.9780898719468.

24 Alexander M Ostrowski. *Solution of Equations and Systems of Equations: Pure and Applied Mathematics: A Series of Monographs and Textbooks, Vol. 9*, volume 9. Elsevier, 2016.

25 Michael J. Todd. *The computation of fixed points and applications*, volume 124. Lecture Notes in Economics and Mathematical Systems, 1976. doi:10.1007/978-3-642-50327-6.

26 Nakul Verma. A note on random projections for preserving paths on a manifold. Technical Report Tech. Report CS2011-0971, UC San Diego, 2011.

27 Rephael Wenger. *Isosurfaces: geometry, topology, and algorithms*. AK Peters/CRC Press, 2013.

28 H. Whitney. *Geometric Integration Theory*. Princeton University Press, 1957. doi:10.1515/9781400877577.

Translating Hausdorff Is Hard: Fine-Grained Lower Bounds for Hausdorff Distance Under Translation

Karl Bringmann
Universität des Saarlandes, Saarbrücken, Germany
Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany

André Nusser
Saarbrücken Graduate School of Computer Science, Universität des Saarlandes, Germany
Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany

Abstract

Computing the similarity of two point sets is a ubiquitous task in medical imaging, geometric shape comparison, trajectory analysis, and many more settings. Arguably the most basic distance measure for this task is the Hausdorff distance, which assigns to each point from one set the closest point in the other set and then evaluates the maximum distance of any assigned pair. A drawback is that this distance measure is not translational invariant, that is, comparing two objects just according to their shape while disregarding their position in space is impossible.

Fortunately, there is a canonical translational invariant version, the Hausdorff distance under translation, which minimizes the Hausdorff distance over all translations of one of the point sets. For point sets of size n and m, the Hausdorff distance under translation can be computed in time $\tilde{\mathcal{O}}(nm)$ for the L_1 and L_∞ norm [Chew, Kedem SWAT'92] and $\tilde{\mathcal{O}}(nm(n+m))$ for the L_2 norm [Huttenlocher, Kedem, Sharir DCG'93].

As these bounds have not been improved for over 25 years, in this paper we approach the Hausdorff distance under translation from the perspective of fine-grained complexity theory. We show (i) a matching lower bound of $(nm)^{1-o(1)}$ for L_1 and L_∞ assuming the Orthogonal Vectors Hypothesis and (ii) a matching lower bound of $n^{2-o(1)}$ for L_2 in the imbalanced case of $m = \mathcal{O}(1)$ assuming the 3SUM Hypothesis.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness

Keywords and phrases Hausdorff Distance Under Translation, Fine-Grained Complexity Theory, Lower Bounds

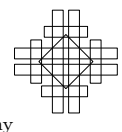
Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.18

Funding *Karl Bringmann*: This work is part of the project TIPEA that has received funding from the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation programme (grant agreement No. 850979).

1 Introduction

As data sets become larger and larger, the requirement for faster algorithms to handle such amounts of data becomes increasingly necessary. One very common type of data that is created during measurements is point sets in the plane, for example when recording GPS trajectories or describing shapes of objects, in medical image analysis, and in various data science applications.

A fundamental algorithmic tool for analyzing point sets is to compute the similarity of two given sets of points. There are several different measures of similarity in this setting, for example Hausdorff distance [21], geometric bottleneck matching [18], Fréchet distance [3], and Dynamic Time Warping [25]. Among these measures, the Hausdorff distance is arguably



the most basic and intuitive: It assigns to each point from one set the closest point in the other set and then evaluates the maximum distance of all assigned pairs of points.[1] For a discussion of the other previously mentioned distance measures, see Section 1.1.

While these similarity measures are of great practical relevance, for some applications it is a drawback that they are not translational invariant, i.e., when translating a point set the distance can – and in most cases will – change. This is unfavorable in applications that ask for comparing the shape of two objects, meaning that the absolute position of an object is irrelevant. Examples of this task arise for example in 2D object shape similarity, medical image analysis [19], classification of handwritten characters [10], movement patterns of animals [12], and sports analysis [17].

Fortunately, any point set similarity measure has a canonical translational invariant version, by minimizing the similarity measure over all translations of the two given point sets. For the Hausdorff distance this variant is known as the *Hausdorff distance under translation*, see Section 2 for a formal definition. Given two point sets in the plane of size n and m, the Hausdorff distance under translation can be computed in time $\mathcal{O}(nm \log^2 nm)$ for the L_1 and L_∞ norm [16], and in time $\mathcal{O}(nm(n+m) \log nm)$ for the L_2 norm [22]. We are not aware of any lower bounds for this problem, not even conditional on a plausible hypothesis. The only results in this direction are $\Omega(n^3)$ lower bounds on the arrangement size [16] and on the number of connected components of the feasible translations [28] (for the decision problem on points in the plane with $n = m$). However, these bounds also hold for L_1 and L_∞, where they are "broken" by the $\mathcal{O}(nm \log^2 nm)$-time algorithm [16], so apparently these bounds are irrelevant for the running time complexity.

In this paper, we approach the Hausdorff distance under translation from the viewpoint of fine-grained complexity theory [29]. For two problem settings, we show that the known algorithms are optimal up to lower order factors assuming standard hypotheses:

1. We show an $(nm)^{1-o(1)}$ lower bound for L_1 and L_∞, matching the $\mathcal{O}(nm \log^2 nm)$-time algorithm from [16] up to lower order factors.

   This result holds conditional on the Orthogonal Vectors Hypothesis, which states that finding two orthogonal vectors among two given sets of n binary vectors in d dimensions cannot be done in time $\mathcal{O}(n^{2-\varepsilon}\text{poly}(d))$ for any $\varepsilon > 0$. It is well-known that the Orthogonal Vectors Hypothesis is implied by the Strong Exponential Time Hypothesis [30], and thus our lower bound also holds assuming the latter [23]. These two hypotheses are the most standard assumptions used in fine-grained complexity theory in the last decade [29].
2. We show an $n^{2-o(1)}$ lower bound for L_2 in the imbalanced case $m = \mathcal{O}(1)$, matching the $\mathcal{O}(nm(n+m) \log nm)$-time algorithm from [16] up to lower order factors. Previously, an $n^{2-o(1)}$ lower bound was only known for the more general problem of computing the Hausdorff distance under translation of sets of *segments* in the case that both sets have size n (a problem for which the best known algorithm runs in time[2] $\tilde{\mathcal{O}}(n^4)$) [6].

   Our result holds conditional on the 3SUM Hypothesis, which states that deciding whether among n given integers there are three that sum up to 0 requires time $n^{2-o(1)}$. This hypothesis was introduced by Gajentaan and Overmars [20], is a standard assumption in computational geometry [24], and has also found a wealth of applications beyond geometry (see, e.g., [26, 4, 2, 1]).

[1] There is a directed and an undirected variant of the Hausdorff distance, see Section 2. In this introduction, we do not differentiate between these two, since all our statements hold for both variants.

[2] By $\tilde{\mathcal{O}}$-notation we ignore logarithmic factors in n and m.

Our lower bounds close gaps that have not seen any progress over 25 years. Furthermore, note that our second lower bound shows a separation between the L_2 norm and the L_1 and L_∞ norms, as in the imbalanced case $m = \mathcal{O}(1)$ the former admits a $\tilde{\mathcal{O}}(n)$-time algorithm [16] while the latter requires time $n^{2-o(1)}$ assuming the 3SUM Hypothesis. We leave it as an open problem whether for L_2 the balanced case $n = m$ requires time $n^{3-o(1)}$.

1.1 Related work

Our work continues a line of research on fine-grained lower bounds in computational geometry, which had early success with the 3SUM Hypothesis [20] and recently got a new impulse with the Orthogonal Vectors Hypothesis (or Strong Exponential Time Hypothesis) and resulting lower bounds for the Fréchet distance [7], see also [13, 11]. Continuing this line of research is getting increasingly difficult, although there are still many classic problems from computational geometry without matching lower bounds. In this paper we obtain such bounds for two settings of the classic Hausdorff distance under translation.

Besides Hausdorff distance, there are several other distance measures on point sets, including geometric bottleneck matching [18], Fréchet distance [3], and Dynamic Time Warping [25]. The geometric bottleneck matching minimizes the maximal distance in a perfect matching between the two given point sets. Fréchet distance and Dynamic Time Warping additionally take the order of the input points into account. They both consider the same class of *traversals* of the input points, and the Fréchet distance minimizes the *maximal* distance that occurs during the traversal, while Dynamic Time Warping minimizes the *sum* of distances.

Let us discuss the canonical translational invariant versions of these distance measures. For geometric bottleneck matching under translation, Efrat et al. designed an $\tilde{\mathcal{O}}(n^5)$ algorithm [18]. The discrete Fréchet distance under translation has an $\tilde{\mathcal{O}}(n^{4.66\cdots})$-time algorithm and a conditional lower bound of $n^{4-o(1)}$ [9], see also [10] for algorithm engineering work on this topic. While Dynamic Time Warping is a very popular measure (in particular for video and speech processing), no exact algorithm for its canonical translational invariant version is known in L_2 since it contains the geometric median problem as a special case [5].

Further work on the Hausdorff distance under translation includes an $\mathcal{O}((n+m)\log nm)$-time algorithm for point sets in one dimension [27]. For generalizations to dimensions $d > 2$ see [16, 15].

2 Preliminaries

In this paper, we consider finite point sets which lie in $\mathbb{R}^2$. For any $p \in \mathbb{R}^2$, we use p_x and p_y to refer to its first and second component, respectively. For a point set $A \subset \mathbb{R}^2$ and a translation $\tau \in \mathbb{R}^2$, we define $A + \tau \coloneqq \{a + \tau \mid a \in A\}$. To denote index sets, we often use $[n] \coloneqq \{1, \dots, n\}$. Given a point $x \in \mathbb{R}^2$, its p-norm is defined as

$$\|x\|_p \coloneqq \left(\sum_{i \in [d]} |x_i|^p \right)^{\frac{1}{p}}.$$

We now introduce several distance measures, which are all versions of the famous Hausdorff distance. First, let us define the most basic version. Let $A, B \subset \mathbb{R}^2$ be two point sets. The *directed Hausdorff distance* is defined as

$$\delta_{\vec{H}}(A, B) \coloneqq \max_{a \in A} \min_{b \in B} \|a - b\|_p.$$

Note that, intuitively, the directed Hausdorff distance measures the distance from A to B but not from B to A, and it is not symmetric. A symmetric variant of the Hausdorff distance, the *undirected Hausdorff distance*, is defined as

$$\delta_H(A,B) \coloneqq \max\{\delta_{\vec{H}}(A,B), \delta_{\vec{H}}(B,A)\}.$$

Note that, by definition, $\delta_{\vec{H}}(A,B) \leq \delta_H(A,B)$. Both of the above distance measures can be modified to a version which is invariant under translation. The *directed Hausdorff distance under translation* is defined as

$$\delta^T_{\vec{H}}(A,B) \coloneqq \min_{\tau\in\mathbb{R}^2} \delta_{\vec{H}}(A, B+\tau),$$

and the *undirected Hausdorff distance under translation* is defined as

$$\delta^T_H(A,B) \coloneqq \min_{\tau\in\mathbb{R}^2} \delta_H(A, B+\tau).$$

Again, it holds that $\delta^T_{\vec{H}}(A,B) \leq \delta^T_H(A,B)$. Naturally, for all of the above distance measures, the decision problem is defined such that we are given two point sets A, B and a threshold distance δ, and ask if the distance of A, B is at most δ.

For the Hausdorff distance (without translation) the undirected distance is at most as hard as the directed distance, because the undirected distance can be calculated using two calls to an algorithm computing the directed distance.[3] However, note that for the Hausdorff distance under translation, we cannot just compute the directed distance twice and then obtain the undirected distance as we have to take the maximum for the same translation.

3 OV based $(mn)^{1-o(1)}$ lower bound for L_1 and L_∞

We now present a conditional lower bound of $(mn)^{1-o(1)}$ for the Hausdorff distance under translation for L_1 and L_∞. For simplicity, we present the lower bound for the L_1 case. This construction is equivalent to the L_∞ case, via a rotation by $\frac{\pi}{4}$. Our lower bound is based on the hypothesized hardness of the Orthogonal Vectors problem.

▶ **Definition 1** (Orthogonal Vectors Problem (OV)). *Given two sets $X, Y \subset \{0,1\}^d$ with $|X| = m, |Y| = n$, decide whether there exist $x \in X$ and $y \in Y$ with $\langle x, y\rangle = 0$.*

A popular hypothesis from fine-grained complexity theory is as follows.

▶ **Definition 2** (Orthogonal Vectors Hypothesis (OVH)). *The Orthogonal Vectors problem cannot be solved in time $\mathcal{O}((nm)^{1-\epsilon}\text{poly}(d))$ for any $\epsilon > 0$.*

This hypothesis is typically stated and used for the balanced case $n = m$. However, it is known that the hypothesis for the balanced case is equivalent to the hypothesis for any unbalanced case $n = m^\alpha$ for any fixed constant $\alpha > 0$, see, e.g, [8, Lemma 5.1 in Arxiv version].

We now describe a reduction from Orthogonal Vectors to Hausdorff distance under translation. To this end, we are given two sets of d-dimensional binary vectors $X = \{x_1, \ldots, x_m\}$ and $Y = \{y_1, \ldots, y_n\}$ with $|X| = m$ and $|Y| = n$, and we construct an instance of the undirected Hausdorff distance under translation defined by point sets A and B and a

[3] Actually, the directed Hausdorff distance is also at most as hard as the undirected Hausdorff distance (thus, they are equally hard), as $\delta_{\vec{H}}(A,B) = \delta_H(A \cup B, B)$.

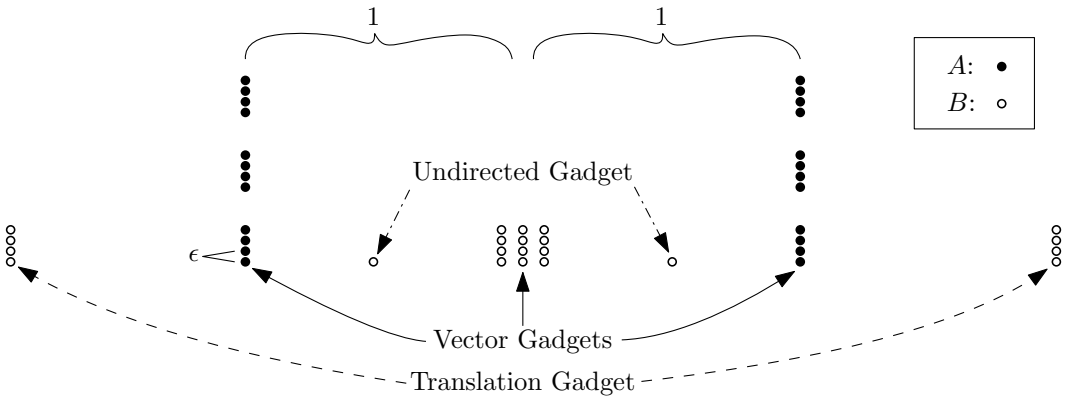


Figure 1 Sketch of the reduction from OV to the undirected Hausdorff distance under translation. The microtranslations in the order of ϵ^2 are not shown in this sketch.

decision distance $\delta = 1$. First, we describe the high-level structure of our reduction. The point set A consists only of Vector Gadgets, which encode the vectors of X using $2md$ points. The point set B consists of three types of gadgets:

- *Vector Gadgets:* They encode the vectors from Y, very similar to the Vector Gadgets of A.
- *Translation Gadget:* It restricts the possible translations of the point set B.
- *Undirected Gadget:* It makes our reduction work for the undirected Hausdorff distance under translation by ensuring that the maximum over the directed Hausdorff distances is always attained by $\delta_{\vec{H}}(B + \tau, A)$.

Intuitively, the two dimensions of the translation choose the vectors $x \in X$ and $y \in Y$ by aligning a Vector Gadget from A with a Vector Gadget from B in a certain way. An alignment of distance at most 1 is only possible if x and y are orthogonal. See Figure 1 for an overview of the reduction.

3.1 Gadgets

We now describe the gadgets in detail. Let $\epsilon > 0$ be a sufficiently small constant, e.g., think of $\epsilon = \frac{1}{20mnd}$. Recall that the distance for which we want to solve the decision problem is $\delta = 1$. Furthermore, we denote the ith component of a vector v by $v[i]$.

Vector Gadget

We define a general Vector Gadget, which we then use at several places by translating it. Given a vector $v \in \{0,1\}^d$, the Vector Gadget consists of the points $p_1, \ldots, p_d \in \mathbb{R}^2$:

$$p_i = \begin{cases} (\epsilon^2, i\epsilon), & \text{if } v[i] = 0 \\ (0, i\epsilon), & \text{if } v[i] = 1 \end{cases}$$

We denote the Vector Gadget created from vector v by $V(v)$. Additionally, we define a mirrored version of the gadget $V(v)$, defined as

$$\overline{V} := V(\bar{v}),$$

where $\bar{v}$ is the inversion of v, i.e., each bit is flipped.

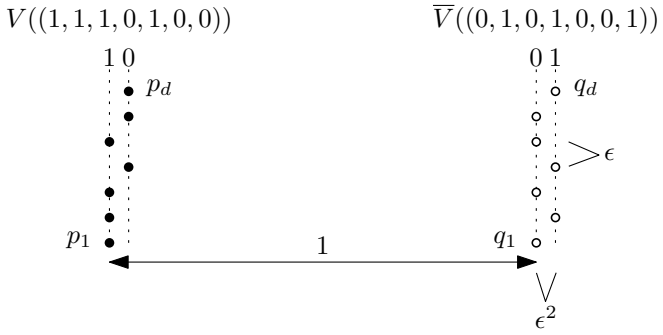


Figure 2 A depiction of the two types of Vector Gadgets and how they are placed to check for orthogonality.

▶ **Lemma 3.** *Given two vectors $v_1, v_2 \in \{0,1\}^d$ and corresponding Vector Gadgets $V_1 = V(v_1)$ and $V_2 = \overline{V}(v_2) + (1,0)$, $\delta_H(V_1, V_2) \leq 1$ if and only if $v_1 \cdot v_2 = 0$.*

Proof. Let the points of V_1 (resp. V_2) be denoted as $p_1, \ldots, p_d$ (resp. $q_1, \ldots, q_d$). First, note that $\|p_i - q_j\|_1 = 1 + |i-j|\epsilon + (v_1[i] + v_2[j] - 1)\epsilon^2 > 1$ for $i \neq j$. Thus, for the Hausdorff distance to be at most 1, we have to match p_i to q_i for all $i \in [d]$. This is possible if and only if $v_1[i] = 0$ or $v_2[i] = 0$, as p_i and q_i are only far for $v_1[i] = 1$ and $v_2[i] = 1$. ◀

See Figure 2 for an example. Note that if we swap both gadgets and invert both vectors (i.e., flip all their bits), the Hausdorff distance does not change and thus an analogous version of Lemma 3 holds in this case, as we are just performing a double inversion.

▶ **Lemma 4.** *Given two vectors $v_1, v_2 \in \{0,1\}^d$ and corresponding Vector Gadgets $V_1 = \overline{V}(v_1)$ and $V_2 = V(v_2)+(1,0)$, $\delta_H(V_1, V_2) \leq 1$ if and only if $\bar{v}_1 \cdot \bar{v}_2 = 0$, where $\bar{v}_1, \bar{v}_2$ are the inversions of v_1, v_2.*

For two Vector Gadgets $V_1 = V(v_1) + (x, y)$ and $V_2 = \overline{V}(v_2) + (x + D, y)$, we say that V_1 and V_2 are *vertically aligned*, or more precisely *vertically aligned in distance D*.

Translation Gadget

To ensure that B cannot be translated arbitrarily, we introduce a gadget to restrict the translations to the regime we require. The Translation Gadget T consists of two translated Vector Gadgets of the zero vector:

$$T \coloneqq (\overline{V}(1^d) - (2 + n\epsilon, 0)) \cup (\overline{V}(0^d) + (2 + 2\epsilon, 0))$$

We show that a simple property on the other set involved in the Hausdorff distance under translation instance already restricts the feasible translations significantly.

▶ **Lemma 5.** *Let $P \subset [-1 - \frac{1}{2}\epsilon, 1 + \frac{1}{2}\epsilon] \times \mathbb{R}$ be a point set. If $\delta_{\vec{H}}^T(T, P) \leq 1$, then $\tau_x^* \in [-(n + \frac{1}{2})\epsilon - \epsilon^2, -\frac{3}{2}\epsilon]$, where τ^* is any translation satisfying $\delta_{\vec{H}}(T, P + \tau^*) \leq 1$.*

Proof. We show the contrapositive. Therefore, assume the converse, i.e., that $\tau_x^* \notin [-(n + \frac{1}{2})\epsilon - \epsilon^2, -\frac{3}{2}\epsilon]$. If $\tau_x^* < -(n + \frac{1}{2})\epsilon - \epsilon^2$, then $-1 - \frac{1}{2}\epsilon - (-2 + n\epsilon + \epsilon^2 + \tau_x^*) > 1$ and thus the left part of T cannot contain any point of P in distance 1. If $\tau_x^* > -\frac{3}{2}\epsilon$, then $2 + 2\epsilon + \tau_x^* - (1 + \frac{1}{2}\epsilon) > 1$ and thus the right part of T cannot contain any point of P in distance 1. Thus, $\delta_{\vec{H}}^T(T, P) > 1$. ◀

Undirected Gadget

To ensure that each point in A can be matched to a point in B with distance at most 1, we add auxiliary points to B. The Undirected Gadget is defined by the point set

$$U := \{(-\frac{1}{2}, 0), (\frac{1}{2}, 0)\}.$$

▶ **Lemma 6.** *Given a set of points $P \subset [-1-\frac{1}{2}\epsilon, 1+\frac{1}{2}\epsilon]\times[-\frac{1}{8}, \frac{1}{8}]$, it holds that $\delta_{\vec{H}}(P, U+\tau) \leq 1$ for any $\tau \in [-(n+\frac{1}{2})\epsilon - \epsilon^2, (n+\frac{1}{2})\epsilon + \epsilon^2] \times [-\frac{1}{8}, \frac{1}{8}]$.*

Proof. By symmetry, we can restrict to proving that the distance of the point set

$$P' = P \cap [0, (n+\frac{1}{2})\epsilon + \epsilon^2] \times [-\frac{1}{8}, \frac{1}{8}]$$

to $(\frac{1}{2}, 0)+\tau$ is at most 1. For any $p' \in P'$, we have $|p'_x - (\frac{1}{2}+\tau_x)| \leq \frac{1}{2} + \mathcal{O}(n\epsilon)$ and $|p'_y - \tau_y| \leq \frac{1}{4}$. Thus, $\|p' - ((\frac{1}{2}, 0) + \tau)\|_1 = \frac{3}{4} + \mathcal{O}(n\epsilon)$, which is less than 1 for small enough ϵ. ◀

3.2 Reduction and correctness

We now describe the reduction and prove its correctness. We construct the point sets of our Hausdorff distance under translation instance as follows. The first set, i.e., set A, consists only of Vector Gadgets:

$$A := \left(\bigcup_{i\in[m]} V(x_i) + (-1 - \frac{1}{2}\epsilon, i \cdot 2d\epsilon)\right) \cup \left(\bigcup_{i\in[m]} V(1^d) + (1 + \frac{1}{2}\epsilon, i \cdot 2d\epsilon)\right)$$

The second set, i.e., set B, consists of Vector Gadgets, the Translation Gadget, and the Undirected Gadget:

$$B := \left(\bigcup_{j\in[n]} \overline{V}(y_j) + (j\epsilon, 0)\right) \cup T \cup U$$

See Figure 1 for a sketch of the above construction. To reference the vector gadgets as they are used in the reduction, we use the notation

$$V_r(x_i) := V(x_i) + (-1 - \frac{1}{2}\epsilon, i \cdot 2d\epsilon) \quad \text{and} \quad \overline{V}_r(y_j) := \overline{V}(y_j) + (j\epsilon, 0).$$

We can now prove correctness of our reduction. In the reduction, we return some canonical positive instance, if the 0^d vector is contained in any of the two OV sets. This allows us to drop all 1^d vectors from the input, as they cannot be orthogonal to any other vector. Thus, we can assume that all vectors in our input contain at least one 0-entry and at least one 1-entry.

▶ **Theorem 7.** *Computing the directed or undirected Hausdorff distance under translation in L_1 or L_∞ for two sets of size n and m cannot be solved in time $\mathcal{O}((mn)^{1-\gamma})$ for any $\gamma > 0$, unless the Orthogonal Vectors Hypothesis fails.*

Proof. Recall that we only have to consider the L_1 case. We first prove that there is a pair of orthogonal vectors $x \in X$ and $y \in Y$ if and only if $\delta_H^T(A, B) \leq 1$.

$\Rightarrow$: Assume that there exist $x_i \in X$, $y_j \in Y$ and $\langle x_i, y_j \rangle = 0$. Then consider the translation $\tau = (-(j+\frac{1}{2})\epsilon, i \cdot 2d\epsilon)$ which vertically aligns the Vector Gadgets $V_r(x_i)$ and $\overline{V}_r(y_j)+\tau$ in distance 1. As x_i and y_j are orthogonal, it follows from Lemma 3 that $\delta_{\vec{H}}(\overline{V}_r(y_j)+\tau, A) \leq 1$. It remains to show that all remaining points of $B + \tau$ have a point in distance at most 1. The Vector Gadgets in $B + \tau$ which correspond to $y_{j'}$ with $j' < j$ are strictly to the left of $\overline{V}_r(y_j) + \tau$ and are thus also in Hausdorff distance at most 1 from $V_r(x_i)$. If $j = n$, then we are done with the Vector Gadgets. Otherwise, consider the Vector Gadget $\overline{V}_r(y_{j+1}) + \tau$. We claim that each point of it is in distance at most 1 from $V(1^d) + (1 + \frac{1}{2}\epsilon, i \cdot 2d\epsilon)$. As the two gadgets are vertically aligned, we just have to check their horizontal distance, which is

$$1 + \frac{1}{2}\epsilon - ((j+1)\epsilon - (j + \frac{1}{2})\epsilon) = 1.$$

Thus, by Lemma 3, we have $\delta_{\vec{H}}(\overline{V}_r(y_{j+1}) + \tau, A) \leq 1$. Now, by the same argument as above, all gadgets $\overline{V}_r(y_{j'}) + \tau$ with $j' > j + 1$ are in directed Hausdorff distance at most 1 from A.
As the points of the Undirected Gadget $U + \tau$ are closer by a distance of almost $\frac{1}{2}$ to A than the Vector Gadgets in $B + \tau$, also $\delta_{\vec{H}}(U + \tau, A) \leq 1$ holds. Finally, we have to show that the Translation Gadget $T + \tau$ is in distance at most 1 from A. As the left part of T and $V_r(x_i)$ are aligned vertically, we only have to check the horizontal distance. The horizontal distance is

$$-1 - \frac{1}{2}\epsilon - (-2 + n\epsilon - (j + \frac{1}{2})\epsilon) = 1 - (n - j)\epsilon \leq 1$$

for any $j \in [n]$. Similarly, the distance of the right part of the Translation Gadget from the vertically aligned $V(1^d)$ in A is

$$2 + 2\epsilon - (j + \frac{1}{2})\epsilon - (1 + \frac{1}{2}\epsilon) = 1 - (j - 1)\epsilon \leq 1$$

for any $j \in [n]$. Thus, by Lemma 3 and Lemma 4, it holds that $\delta_{\vec{H}}(T + \tau, A) \leq 1$. As $\tau \in [-(n + \frac{1}{2})\epsilon - \epsilon^2, -\frac{3}{2}\epsilon] \times [-\frac{1}{8}, \frac{1}{8}]$, we know by Lemma 6 that $\delta_{\vec{H}}(A, B + \tau) \leq 1$ and thus also $\delta_H^T(A, B) \leq 1$.

$\Leftarrow$: Now, assume that $\delta_H^T(A, B) \leq 1$ and let τ be any translation for which $\delta_{\vec{H}}(B + \tau, A) \leq 1$. We used the directed Hausdorff distance in the previous statement on purpose, as we prove hardness for both versions. Lemma 5 implies that $\tau_x \in [-(n + \frac{1}{2})\epsilon - \epsilon^2, -\frac{3}{2}\epsilon]$.
Let $\overline{V}_r(y_j) + \tau, \overline{V}_r(y_{j+1}) + \tau$ be the Vector Gadgets such that $\overline{V}_r(y_j) + \tau$ has directed Hausdorff distance at most 1 to the left Vector Gadgets of A and $\overline{V}_r(y_{j+1})+\tau$ has directed Hausdorff distance at most 1 to the right Vector Gadgets of A. This is well-defined as the left Vector Gadgets of A and the right Vector Gadgets of A are in distance at least $2+\epsilon-\epsilon^2$ from each other, and thus no Vector Gadget of $B+\tau$ can be in distance at most 1 from both sides. Furthermore, as $\tau_x \leq -\frac{3}{2}\epsilon$, there has to be a Vector Gadget $\overline{V}_r(y_j)+\tau$ that has directed Hausdorff distance at most 1 to the left Vector Gadgets of A, as

$$j\epsilon - \frac{3}{2}\epsilon - (-1 - \frac{1}{2}\epsilon) = 1 + (j - 1)\epsilon \leq 1$$

for $j = 1$. If $j = n$, then $\overline{V}_r(y_{j+1}) + \tau$ is undefined.

As $\delta_{\vec{H}}(B+\tau, A) \leq 1$, we know that $\overline{V}_r(y_j)+\tau$ has directed Hausdorff distance at most 1 to a gadget $V_r(x)$ for some $x \in X$. We claim that this distance cannot be closer than 1 as $\overline{V}_r(y_{j+1})+\tau$ must have a directed Hausdorff distance at most 1 from the right side of A or, in case $j=n$, due to the restrictions imposed by the Translation Gadget. Let us consider the case $j \neq n$ first. Any translation τ' which places $\overline{V}_r(y_{j+1})+\tau'$ in directed Hausdorff distance at most 1 from the right side of A needs to fulfill

$$1+\frac{1}{2}\epsilon-((j+1)\epsilon+\tau_1') \leq 1$$

and thus $\tau_1' \geq -(j+\frac{1}{2})\epsilon$, using the fact that each vector in Y contains at least one 0-entry. This, on the other hand, implies that $\overline{V}_r(y_j)+\tau'$ is in Hausdorff distance at least

$$j\epsilon-(j+\frac{1}{2})\epsilon-(-1-\frac{1}{2}\epsilon)=1$$

from $V_r(x)$. Now consider the case $j=n$. As by Lemma 5 we have $\tau_x \geq -(n+\frac{1}{2})\epsilon-\epsilon^2$, it follows that $\overline{V}_r(y_n)+\tau$ is in Hausdorff distance at least

$$n\epsilon-(n+\frac{1}{2})\epsilon-(-1-\frac{1}{2}\epsilon)=1$$

from $V_r(x)$, using the fact that each vector in Y contains at least one 0-entry (this is the reason why the ϵ^2 disappears).
By the arguments above, the two gadgets $\overline{V}_r(y_j)+\tau$ and $V_r(x)$ have to be horizontally aligned as required by Lemma 3. They also have to be vertically aligned as a vertical deviation would incur a Hausdorff distance larger than 1 for the pair of points in the two gadgets that are in horizontal distance 1. Then, applying Lemma 3, it follows that x an y_j are orthogonal.

It remains to argue why the above reduction implies the lower bound stated in the theorem. Assume we have an algorithm that computes the Hausdorff distance under translation for L_1 or L_∞ in time $(mn)^{1-\gamma}$ for some $\gamma > 0$. Then, given an Orthogonal Vectors instance X, Y with $|X|=m$ and $|Y|=n$, we can use the described reduction to obtain an equivalent Hausdorff under translation instance with point sets A, B of size $|A|=\mathcal{O}(md)$ and $|B|=\mathcal{O}(nd)$ and solve it in time $\mathcal{O}((mn)^{1-\gamma}\text{poly}(d))$, contradicting the Orthogonal Vectors Hypothesis. ◀

3.3 Generalization to L_p

We believe that we can extend the above construction such that it works for all L_p norms with $p \neq \infty$ by changing the spacing between 0 and 1 points of the Vector Gadgets and also set ϵ accordingly. More precisely, it seems that we can use ϵ^{2p} as spacing (instead of ϵ^2) and set $\epsilon < \frac{1}{40pmnd}$. The proofs should then be analogous to the L_1 case.

4 3Sum based $n^{2-o(1)}$ lower bound for $m \in \mathcal{O}(1)$

We now present a hardness result for the unbalanced case of the directed and undirected Hausdorff distance under translation. We base our hardness on another popular hypothesis of fined-grained complexity theory: the 3Sum Hypothesis. Before stating the hypothesis, let us first introduce the 3Sum problem.

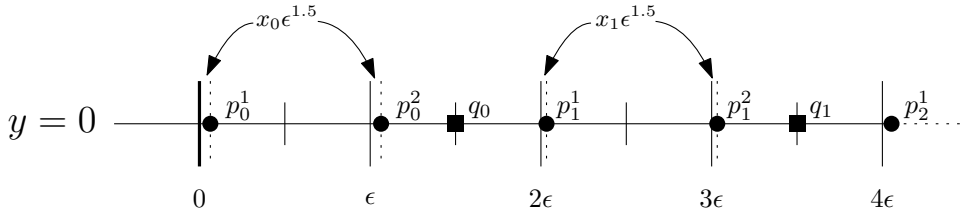


Figure 3 The A set of the low-level gadget of the 3Sum reduction, which is used to build the high-level gadgets. We just show the leftmost part of the gadget, but the remainder is similar.

▶ **Definition 8** (3Sum). *Given three sets of positive integers X, Y, Z all of size n, do there exist $x \in X, y \in Y, z \in Z$ such that $x + y = z$?*

The corresponding hardness assumption is the 3Sum Hypothesis.

▶ **Definition 9** (3Sum Hypothesis). *There is no $\mathcal{O}(n^{2-\epsilon})$ algorithm for 3Sum for any $\epsilon > 0$.*

There are several equivalent variants of the 3Sum problem. Most important for us is the convolution 3Sum problem, abbreviated as Conv3Sum [26, 14].

▶ **Definition 10** (Conv3SUM). *Given a sequence of positive integers $X = (x_1, \ldots, x_n)$ of size n, do there exist i, j such that $x_i + x_j = x_{i+j}$?*

This problem has a trivial $\mathcal{O}(n^2)$ algorithm and, assuming the 3Sum Hypothesis, this is also optimal up to lower order factors. As 3Sum and Conv3Sum are equivalent, a lower bound conditional on Conv3Sum implies a lower bound conditional on 3Sum.

Therefore, given a Conv3Sum instance defined by the set of integers X with $|X| = n$, we create an equivalent instance of the directed Hausdorff distance under translation for L_2 by constructing two sets of points A and B with $|A| = \mathcal{O}(n)$ and $|B| = \mathcal{O}(1)$ and providing a decision distance δ. Intuitively, we define a low-level gadget from which we build three high-level gadgets by rotation and scaling. Recall that in the Conv3Sum problem we have to find values i, j which fulfill the equation $x_i + x_j = x_{i+j}$. We encode the choice of these two values into the two dimensions of the translation. These three high-level gadgets then verify if the Conv3Sum equation is fulfilled. In the remainder of this section, we present the details of our reduction and prove that it implies the claimed lower bound.

4.1 Construction

Given an integer Conv3Sum instance with $X \subset [M]$ where $n = |X|$, we now describe the construction of the Hausdorff distance under translation instance with point sets A, B and threshold distance δ. We use a small enough ϵ, e.g., $\epsilon = (4Mn^2)^{-2}$, as value for microtranslations. Furthermore, we set $\delta = 1+4n^2\epsilon^2$. The additional $4n^2\epsilon^2$ term compensates for the small variations in distance that occur on microtranslations due to the curvature of the L_2-ball.

4.1.1 Low-level gadget

We use a single low-level gadget, which is then scaled and rotated to obtain high-level gadgets. This gadget consists of two point sets A_l and B_l. The point set A_l contains what we call *number points* p_i^1, p_i^2 and *filling points* q_i for $0 \leq i < n$. The set B_l just contains two points:

r_1 and r_2. The number points p_i^1, p_i^2 encode the number x_i, while the filling points make sure that no other translations than the desired ones are possible. See Figure 3 for an overview. All of the points in this gadget are of the form $(x, 0)$. The number points are

$$p_i^1 = \left(2i\epsilon + x_i\epsilon^{1.5}, 0\right), \quad p_i^2 = p_i^1 + (\epsilon, 0)$$

for $0 \leq i < n$. The filling points are

$$q_i = \left(\left(2i + \frac{3}{2}\right)\epsilon, 0\right)$$

for $0 \leq i < n$.

The points in B_l should introduce a gap to only allow alignment of the number gadgets such that the microtranslations (i.e., those in the order of $\epsilon^{1.5}$) correspond to the number of the gap in the number gadget. Thus B_l contains the points

$$r_1 = (-1, 0), \quad r_2 = (1 + \epsilon, 0).$$

Before we prove properties of the low-level gadget, we first prove that the error that is happening due to the curvature of the L_2-ball is small.

▶ **Lemma 11.** *Let $(p_x, p_y), (q_x, q_y) \in \mathbb{R}^2$ be two points with $|p_x - q_x| \in [\frac{1}{2}, 2]$ and $p_y = q_y$. For any $\tau \in [0, (2n-1)\epsilon]^2$, we have*

$$|p_x - (q_x + \tau_x)| \leq \|p - (q + \tau)\|_2 \leq |p_x - (q_x + \tau_x)| + 4n^2\epsilon^2.$$

Proof. As each component is a lower bound to the L_2 norm, the first inequality follows. Thus, let us prove the second inequality. We first transform

$$\|p - (q + \tau)\|_2 = \sqrt{(p_x - (q_x + \tau_x))^2 + \tau_y^2} = |p_x - (q_x - \tau_x)|\sqrt{1 + \tau_y^2/(p_x - (q_x + \tau_x))^2}.$$

Because $\sqrt{1+x} \leq 1 + \frac{x}{2}$ for any $x \geq 0$, we have

$$\|p - (q + \tau)\|_2 \leq |p_x - (q_x - \tau_x)| + \tau_y^2/(2|p_x - (q_x - \tau_x)|).$$

As $\tau_y \leq 2(n-1)\epsilon$ and $|p_x - (q_x - \tau_x)| \geq \frac{1}{2}$, we obtain the desired upper bound. ◀

An analogous statement holds when swapping the x and y coordinates. Note that the $4n^2\epsilon^2$ term also occurs in the value of δ that we chose, as this is how we compensate for these errors in our construction. While we have to consider this error in the following arguments, it already seems that it will be insignificant due to its magnitude.

We now state two lemmas which show how the Hausdorff distance under translation decision problem is related to the structure of the low-level gadget.

▶ **Lemma 12.** *Given a low-level gadget A_l, B_l as constructed above and the translation being restricted to $\tau \in [0, (2n-1)\epsilon]^2$, it holds that if $\delta_{\vec{H}}(A_l, B_l + \tau) \leq \delta$, then*

$$\exists i \in \mathbb{N} : \tau_x = 2i\epsilon + x_i\epsilon^{1.5} \pm 4n^2\epsilon^2.$$

Proof. Let $\tau \in [0, (2n-1)\epsilon]^2$ and assume $\delta_{\vec{H}}(A_l, B_l + \tau) \leq \delta$. Then all points in A_l are in distance at most δ from one of the two points in B_l. Furthermore, both points in $B_l + \tau$ also have at least one close point in A_l, as

$$\|r_1+\tau-p_0^1\|_2 \leq 1-\tau_x+4n^2\epsilon^2 \leq \delta \quad \text{and} \quad \|r_2+\tau-q_{n-1}\|_2 \leq 1+\tau_x-(2n-\frac{1}{2})\epsilon+4n^2\epsilon^2 < \delta,$$

using Lemma 11.

The gaps between neighboring points in A_l either have width close to $\frac{1}{2}\epsilon$, if the gap is between a number point and a filling point (p_i^1 and q_{i-1}, or p_i^2 and q_i), or they have a width of ϵ, if the gap is between two number points (p_i^1 and p_i^2). Furthermore, the two points in B_l have distance $2+\epsilon$, so there is an $\epsilon - 8n^2\epsilon^2$ gap between their δ-balls. Thus, there is an i such that p_i^1 has distance at most δ to r_1, and p_i^2 has distance at most δ to r_2. This alignment of the gadgets can only be realized by a translation τ for which

$$\tau_x = 2i\epsilon + x_i\epsilon^{1.5} \pm 4n^2\epsilon^2,$$

which completes the proof. ◀

▶ **Lemma 13.** *Given a low-level gadget A_l, B_l as constructed above and the translation being restricted to $\tau \in [0,(2n-1)\epsilon]^2$, it holds that if*

$$\exists i \in \mathbb{N} : \tau_x = 2i\epsilon + x_i\epsilon^{1.5},$$

then $\delta_H(A_l, B_l+\tau) \leq \delta$.

Proof. Let $i \in \mathbb{N}$ and let $\tau_x = 2i\epsilon + x_i\epsilon^{1.5}$. Consider any translations $\tau \in \{\tau_x\} \times [0, 2(n-1)\epsilon]$. Due to the restricted translation and Lemma 11, we can disregard the error terms that arise from the vertical translation τ_y as they are compensated for by δ. Then all the points in A_l before and including p_i^1 are in distance at most δ from $r_1 \in B_l + \tau$ and all the points afterwards are in distance at most δ from $r_2 \in B_l + \tau$. Clearly, both points in $B_l + \tau$ then also have points from A_l in distance δ, and thus $\delta_H(A_l, B_l+\tau) \leq \delta$. ◀

4.1.2 High-level gadgets

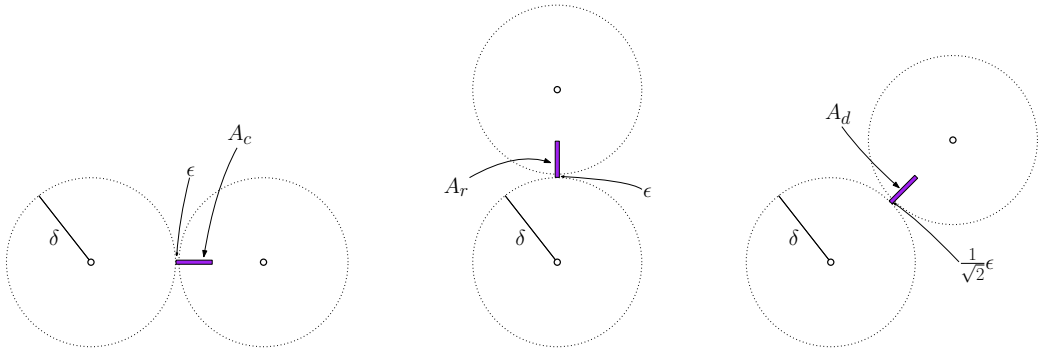
This construction is inspired by the hard instance that was given in [28]. We want to obtain a grid of translations of spacing ϵ with some microtranslations in the $\mathcal{O}(\epsilon^{1.5})$ range. We already defined the low-level gadget above, and we now define the high-level gadgets.

Column Gadget

The column gadget induces columns in translational space, i.e., it enforces that valid translations have to lie on one of these columns. The column gadget is actually the low-level gadget we already described above. You can see a sketch of this gadget in Figure 4a. To semantically distinguish it from the low-level gadget, we refer to the point sets of the column gadget as A_c and B_c.

Row Gadget

The row gadget induces rows in translational space, i.e., it enforces that valid translations have to lie on one of these rows. We obtain the row gadget by rotating all points in the low-level gadget around the origin by $\pi/2$ counterclockwise. You can see a sketch of this gadget in Figure 4b. We call the point sets of the row gadget A_r and B_r.



(a) Column Gadget. **(b)** Row Gadget. **(c)** Diagonal Gadget.

Figure 4 Three of the high-level gadgets. The points of A are all in the low-level gadgets, while the points in B are explicitly shown including their δ-ball.

Diagonal Gadget

The diagonal gadget induces diagonals in translational space, i.e., it enforces that valid translations have to lie on one of these diagonals. As opposed to the column and row gadget, the diagonal gadget also has to be scaled. We scale the sets A_l and B_l separately. We scale A_l such that the gap between the number point pairs p_i^1, p_i^2 becomes $\frac{1}{\sqrt{2}}\epsilon$. And we scale B_l such that the gap between the points becomes $2 + \frac{1}{\sqrt{2}}\epsilon$. After scaling, we rotate the points counterclockwise around the origin by $\pi/4$. You can see a sketch of this gadget in Figure 4c. We call the point sets of the diagonal gadget A_d and B_d.

Translation Gadget

To restrict the translations for the directed Hausdorff distance under translation, we introduce another gadget. The first set of points A_t contains

$$z_l := (-1 + (2n-1)\epsilon, 0), \quad z_r := (1,0), \quad z_b := (0, -1 + (2n-1)\epsilon), \quad z_t := (0,1).$$

The second point set B_t only contains the origin $z_c := (0,0)$. We want to make sure that this gadget behaves well in a certain range.

▶ **Lemma 14.** *Given $\tau \in [0, (2n-1)\epsilon]^2$, it holds that $\delta_H(A_t, B_t + \tau) \leq \delta$.*

Proof. As B_t has a point on all sides, clearly $\delta_{\vec{H}}(B_t + \tau, A_t) \leq \delta$. Furthermore,

$$\|z_l - (z_c + \tau)\|_2 \leq 1 + 4n^2\epsilon^2 \leq \delta \quad \text{and} \quad \|z_r - (z_c + \tau)\|_2 \leq \delta,$$

using Lemma 11. Analogous statements hold for z_b and z_t. Thus, also $\delta_{\vec{H}}(A_t, B_t + \tau) \leq \delta$. ◀

4.1.3 Complete construction

To obtain the final sets of the reduction, we now place all four described high-level gadgets (i.e., column gadget, row gadget, diagonal gadget, and translation gadget) far enough apart. More explicitly, the point sets A, B of the Hausdorff distance under translation instance are defined as

$$A := A_c \cup (A_r + (10,0)) \cup (A_d + (20,0)) \cup (A_t + (30,0))$$

and

$$B := B_c \cup (B_r + (10,0)) \cup (B_d + (20,0)) \cup (B_t + (30,0)).$$

The far placement ensures that the two point sets of the respective gadgets have to be matched to each other for a decision distance δ.

4.2 Proof of correctness

First, we want to ensure that everything relevant happens in a very small range of translations.

▶ **Lemma 15.** *Let $\tau \in \mathbb{R}^2$. If $\delta_{\vec{H}}(A, B + \tau) \leq \delta$, then $\tau \in [0, (2n-1)\epsilon]^2$.*

Proof. Note that for a Hausdorff distance at most δ, the sets A_c and B_c have to matched to each other and analogously for A_r, B_r, and A_d, B_d, and A_t, B_t. To show the contrapositive, now assume $\tau \notin [0, (2n-1)\epsilon]^2$. For simplicity, we refer to the points in the high-level gadgets with the notation of the low-level gadget. Additionally, due to the translation gadget, we have

$$\|z_l - (z_c + \tau)\|_2 > \delta \quad \text{for} \quad \tau_x > (2n-1)\epsilon + 4n^2\epsilon^2,$$

and

$$\|z_r - (z_c + \tau)\|_2 > \delta \quad \text{for} \quad \tau_x < -4n^2\epsilon^2.$$

We now show that under these restricted translations and as $\delta_{\vec{H}}(A, B + \tau) \leq \delta$, both points r_1, r_2 in B_c have at least one point of A_c in distance δ. In the column gadget for $\tau_x \in [-4n^2\epsilon^2, 0)$, we have

$$\|(r_1 + \tau) - p_0^1\|_2 \geq |-1 - (p_0^1)_x + \tau_x| > \delta \quad \text{and} \quad \|(r_2 + \tau) - p_0^1\|_2 \geq 1 + \epsilon - \mathcal{O}(\epsilon^{1.5}) > \delta,$$

for small enough ϵ. On the other hand, for $\tau_x \in ((2n-1)\epsilon, (2n-1)\epsilon + 4n^2\epsilon^2]$, we have

$$\|r_2 + \tau - p_{n-1}^2\|_2 \geq 1 + \epsilon + \tau_x - (2n-1)\epsilon > \delta \quad \text{and} \quad \|r_1 + \tau - p_{n-1}^2\|_2 \geq 1 + \epsilon - \mathcal{O}(\epsilon^{1.5}) > \delta$$

for small enough ϵ. An analogous argument holds for the row gadget and τ_y, as the row gadget is just a rotated version of the column gadget and the translation gadget is symmetric with respect to these gadgets. ◀

We can now prove the main result of this section.

▶ **Theorem 16.** *Computing the directed or undirected Hausdorff distance under translation in L_2 for two sets of size n and $\mathcal{O}(1)$ cannot be solved in time $\mathcal{O}(n^{2-\gamma})$ for any $\gamma > 0$, unless the 3Sum Hypothesis fails.*

Proof. We construct a Hausdorff under translation instance in this proof from a Conv3Sum instance as described previously in this section, and then show that they are equivalent. We first consider how to apply Lemma 12 and Lemma 13 to the diagonal gadget. More precisely, we consider which translations align the gaps of A_d and B_d as is used in these two lemmas. Due to the scaling of the gadget, these translations are of the form $\sqrt{2}\tau_x = 2k\epsilon + x_k\epsilon^{1.5}$. By the rotation, we then obtain translations of the form

$$\frac{\sqrt{2}(\tau_x + \tau_y)}{\sqrt{2}} = \|\tau\|_1 = 2(i+j)\epsilon + x_{i+j}\epsilon^{1.5}.$$

$\Leftarrow$: Assume X is a positive Conv3Sum instance. Then there exist x_i, x_j such that $x_i + x_j = x_{i+j}$. Consider $\tau = (2i\epsilon + x_i\epsilon^{1.5}, 2j\epsilon + x_j\epsilon^{1.5})$ as translation. Due to Lemma 13, we have that $\delta_H(A_c, B_c + \tau) \leq \delta$ and analogously $\delta_H(A_r, B_r + \tau) \leq \delta$. By the initial observation we can also apply Lemma 13 to the diagonal gadget, and thus $\delta_H(A_d, B_d + \tau) \leq \delta$. Finally, by Lemma 14, we also have that $\delta_H(A_t, B_t + \tau) \leq \delta$ for the given τ.

⇒: Assume $\delta_{\bar{H}}^{T}(A,B) \leq \delta$. From Lemma 15, it follows that $\tau \in [0,(2n-1)\epsilon]^2$. Then, due to Lemma 12 and the initial observation about the diagonal gadget, we have that there exist i,j,k that fulfill

$$\tau_x = 2i\epsilon + x_i\epsilon^{1.5} \pm 4n^2\epsilon^2 \quad \text{and} \quad \tau_y = 2j\epsilon + x_j\epsilon^{1.5} \pm 4n^2\epsilon^2 \quad \text{and} \quad \tau_x + \tau_y = 2k\epsilon + x_k\epsilon^{1.5} \pm 4n^2\epsilon^2.$$

It follows that

$$2i\epsilon + x_i\epsilon^{1.5} + 2j\epsilon + x_j\epsilon^{1.5} \pm 8n^2\epsilon^2 = 2k\epsilon + x_k\epsilon^{1.5} \pm 4n^2\epsilon^2,$$

and thus $i+j=k$ and $x_i + x_j = x_k$.

It remains to argue why the above reduction implies the lower bound stated in the theorem. Assume we have an algorithm that computes the Hausdorff distance under translation in L_2 in time $\mathcal{O}(n^{2-\gamma})$ for some $\gamma > 0$. Then, given a Conv3Sum instance X with $|X| = n$, we can use the described reduction to obtain an equivalent Hausdorff under translation instance with point sets A, B of size $|A| = \mathcal{O}(n)$ and $|B| = \mathcal{O}(1)$ and solve it in time $\mathcal{O}(n^{2-\gamma})$, contradicting the 3Sum Hypothesis. ◀

References

1. Amir Abboud, Arturs Backurs, Karl Bringmann, and Marvin Künnemann. Fine-grained complexity of analyzing compressed data: Quantifying improvements over decompress-and-solve. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 192–203. IEEE Computer Society, 2017. doi:10.1109/FOCS.2017.26.
2. Amir Abboud, Virginia Vassilevska Williams, and Oren Weimann. Consequences of faster alignment of sequences. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, volume 8572 of *Lecture Notes in Computer Science*, pages 39–51. Springer, 2014. doi:10.1007/978-3-662-43948-7_4.
3. Helmut Alt and Michael Godau. Computing the Fréchet Distance between Two Polygonal Curves. *Int. J. Comput. Geometry Appl.*, 5:75–91, 1995. doi:10.1142/S0218195995000064.
4. Amihood Amir, Timothy M. Chan, Moshe Lewenstein, and Noa Lewenstein. On hardness of jumbled indexing. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, volume 8572 of *Lecture Notes in Computer Science*, pages 114–125. Springer, 2014. doi:10.1007/978-3-662-43948-7_10.
5. Chanderjit Bajaj. The algebraic degree of geometric optimization problems. *Discrete & Computational Geometry*, 3(2):177–191, 1988.
6. Gill Barequet and Sariel Har-Peled. Polygon containment and translational in-hausdorff-distance between segment sets are 3sum-hard. *International Journal of Computational Geometry & Applications*, 11(04):465–474, 2001. doi:10.1142/S0218195901000596.
7. Karl Bringmann. Why walking the dog takes time: Frechet distance has no strongly subquadratic algorithms unless SETH fails. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 661–670. IEEE Computer Society, 2014. doi:10.1109/FOCS.2014.76.
8. Karl Bringmann and Marvin Künnemann. Multivariate fine-grained complexity of longest common subsequence. In Artur Czumaj, editor, *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 1216–1235. SIAM, 2018. doi:10.1137/1.9781611975031.79.

9 Karl Bringmann, Marvin Künnemann, and André Nusser. Fréchet distance under translation: Conditional hardness and an algorithm via offline dynamic grid reachability. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 2902–2921. SIAM, 2019. doi:10.1137/1.9781611975482.180.

10 Karl Bringmann, Marvin Künnemann, and André Nusser. When lipschitz walks your dog: Algorithm engineering of the discrete fréchet distance under translation. In Fabrizio Grandoni, Grzegorz Herman, and Peter Sanders, editors, *28th Annual European Symposium on Algorithms, ESA 2020, September 7-9, 2020, Pisa, Italy (Virtual Conference)*, volume 173 of *LIPIcs*, pages 25:1–25:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ESA.2020.25.

11 Karl Bringmann and Wolfgang Mulzer. Approximability of the discrete fréchet distance. *J. Comput. Geom.*, 7(2):46–76, 2016. doi:10.20382/jocg.v7i2a4.

12 Kevin Buchin, Anne Driemel, Natasja van de L'Isle, and André Nusser. klcluster: Center-based clustering of trajectories. In Farnoush Banaei Kashani, Goce Trajcevski, Ralf Hartmut Güting, Lars Kulik, and Shawn D. Newsam, editors, *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL 2019, Chicago, IL, USA, November 5-8, 2019*, pages 496–499. ACM, 2019. doi:10.1145/3347146.3359111.

13 Kevin Buchin, Tim Ophelders, and Bettina Speckmann. SETH says: Weak fréchet distance is faster, but only if it is continuous and in one dimension. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 2887–2901. SIAM, 2019. doi:10.1137/1.9781611975482.179.

14 Timothy M. Chan and Qizheng He. Reducing 3sum to convolution-3sum. In Martin Farach-Colton and Inge Li Gørtz, editors, *3rd Symposium on Simplicity in Algorithms, SOSA@SODA 2020, Salt Lake City, UT, USA, January 6-7, 2020*, pages 1–7. SIAM, 2020. doi:10.1137/1.9781611976014.1.

15 L. P. Chew, D. Dor, A. Efrat, and K. Kedem. Geometric pattern matching in d-dimensional space. *Discrete & Computational Geometry*, 21(2):257–274, February 1999. doi:10.1007/PL00009420.

16 L. Paul Chew and Klara Kedem. Improvements on geometric pattern matching problems. In Otto Nurmi and Esko Ukkonen, editors, *Algorithm Theory — SWAT '92*, Lecture Notes in Computer Science, pages 318–325. Springer Berlin Heidelberg, 1992.

17 Mark de Berg, Atlas F. Cook, and Joachim Gudmundsson. Fast fréchet queries. *Computational Geometry*, 46(6):747–755, 2013. doi:10.1016/j.comgeo.2012.11.006.

18 A. Efrat, A. Itai, and M. J. Katz. Geometry Helps in Bottleneck Matching and Related Problems. *Algorithmica*, 31(1):1–28, 2001. doi:10.1007/s00453-001-0016-8.

19 Andriy Fedorov, Eric Billet, Marcel Prastawa, Guido Gerig, Alireza Radmanesh, Simon K Warfield, Ron Kikinis, and Nikos Chrisochoides. Evaluation of brain mri alignment with the robust hausdorff distance measures. In *International Symposium on Visual Computing*, pages 594–603. Springer, 2008.

20 Anka Gajentaan and Mark H. Overmars. On a class of $O(n^2)$ problems in computational geometry. *Comput. Geom.*, 5:165–185, 1995. doi:10.1016/0925-7721(95)00022-2.

21 Felix Hausdorff. *Grundzüge der Mengenlehre*, volume 7. von Veit, 1914.

22 Daniel P Huttenlocher, Klara Kedem, and Micha Sharir. The upper envelope of voronoi surfaces and its applications. *Discrete & Computational Geometry*, 9(3):267–291, 1993.

23 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001. doi:10.1006/jcss.2001.1774.

24 James King. A survey of 3sum-hard problems, 2004.

25 Meinard Müller. *Information retrieval for music and motion*. Springer, 2007. doi:10.1007/978-3-540-74048-3.

26 Mihai Patrascu. Towards polynomial lower bounds for dynamic problems. In *Proceedings of the Forty-Second ACM Symposium on Theory of Computing*, STOC '10, page 603–610, New York, NY, USA, 2010. Association for Computing Machinery. doi:10.1145/1806689.1806772.

27 Günter Rote. Computing the minimum Hausdorff distance between two point sets on a line under translation. *Information Processing Letters*, 38(3):123–127, 1991. doi:10.1016/0020-0190(91)90233-8.

28 W. J. Rucklidge. Lower bounds for the complexity of the graph of the Hausdorff distance as a function of transformation. *Discrete & Computational Geometry*, 16(2):135–153, 1996. doi:10.1007/BF02716804.

29 Virginia Vassilevska Williams. On some fine-grained questions in algorithms and complexity. In *Proc. ICM*, volume 3, pages 3431–3472. World Scientific, 2018.

30 Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theor. Comput. Sci.*, 348(2-3):357–365, 2005. doi:10.1016/j.tcs.2005.09.023.

Optimal Bounds for the Colorful Fractional Helly Theorem

Denys Bulavka
Department of Applied Mathematics, Charles University, Praha, Czech Republic

Afshin Goodarzi
Royal Institute of Technology, Department of Mathematics, Stockholm, Sweden

Martin Tancer
Department of Applied Mathematics, Charles University, Praha, Czech Republic

Abstract

The well known fractional Helly theorem and colorful Helly theorem can be merged into the so called colorful fractional Helly theorem. It states: for every $\alpha \in (0, 1]$ and every non-negative integer d, there is $\beta_{\mathrm{col}} = \beta_{\mathrm{col}}(\alpha, d) \in (0, 1]$ with the following property. Let $\mathcal{F}_1, \dots, \mathcal{F}_{d+1}$ be finite nonempty families of convex sets in $\mathbb{R}^d$ of sizes $n_1, \dots, n_{d+1}$, respectively. If at least $\alpha n_1 n_2 \cdots n_{d+1}$ of the colorful $(d+1)$-tuples have a nonempty intersection, then there is $i \in [d+1]$ such that $\mathcal{F}_i$ contains a subfamily of size at least $\beta_{\mathrm{col}} n_i$ with a nonempty intersection. (A colorful $(d+1)$-tuple is a $(d+1)$-tuple $(F_1, \dots, F_{d+1})$ such that F_i belongs to $\mathcal{F}_i$ for every i.)

The colorful fractional Helly theorem was first stated and proved by Bárány, Fodor, Montejano, Oliveros, and Pór in 2014 with $\beta_{\mathrm{col}} = \alpha/(d+1)$. In 2017 Kim proved the theorem with better function β_{col}, which in particular tends to 1 when α tends to 1. Kim also conjectured what is the optimal bound for $\beta_{\mathrm{col}}(\alpha, d)$ and provided the upper bound example for the optimal bound. The conjectured bound coincides with the optimal bounds for the (non-colorful) fractional Helly theorem proved independently by Eckhoff and Kalai around 1984.

We verify Kim's conjecture by extending Kalai's approach to the colorful scenario. Moreover, we obtain optimal bounds also in a more general setting when we allow several sets of the same color.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases colorful fractional Helly theorem, d-collapsible, exterior algebra, d-representable

Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.19

Related Version *Full Version*: https://arxiv.org/abs/2010.15765

Funding *Denys Bulavka*: Supported by the GAČR grant 19-27871X.
Afshin Goodarzi: Supported by KAW-stipendiet 2015.0360 from the Knut and Alice Wallenberg Foundation.
Martin Tancer: Supported by the GAČR grant 19-04113Y.

Acknowledgements We thank Xavier Goaoc and Eran Nevo for providing us with useful references. We also thank the anonymous referees for useful remarks.

1 Introduction

The target of this paper is to provide optimal bounds for the colorful fractional Helly theorem first stated by Bárány, Fodor, Montejano, Oliveros, and Pór [5], and then improved by Kim [13]. In order to explain the colorful fractional Helly theorem, let us briefly survey the preceding results.

The starting point, as usual in this context, is the Helly theorem:

▶ **Theorem 1** (Helly's theorem [8]). *Let $\mathcal{F}$ be a finite family of at least $d+1$ convex sets in $\mathbb{R}^d$. Assume that every subfamily of $\mathcal{F}$ with exactly $d+1$ members has a nonempty intersection. Then all sets in $\mathcal{F}$ have a nonempty intersection.*

Helly's theorem admits numerous extensions and two of them, important in our context, are the fractional Helly theorem and the colorful Helly theorem. The fractional Helly theorem of Katchalski and Liu covers the case when only some fraction of the $d+1$ tuples in $\mathcal{F}$ has a nonempty intersection.

▶ **Theorem 2** (The fractional Helly theorem [12]). *For every $\alpha \in (0,1]$ and every non-negative integer d, there is $\beta = \beta(\alpha, d) \in (0,1]$ with the following property. Let $\mathcal{F}$ be a finite family of $n \geq d+1$ convex sets in $\mathbb{R}^d$ such that at least $\alpha\binom{n}{d+1}$ of the subfamilies of $\mathcal{F}$ with exactly $d+1$ members have a nonempty intersection. Then there is a subfamily of $\mathcal{F}$ with at least βn members with a nonempty intersection.*

An interesting aspect of the fractional Helly theorem is not only to show the existence of $\beta(\alpha, d)$ but also to provide the largest value of $\beta(\alpha, d)$ with which the theorem is valid. This has been resolved independently by Eckhoff [7] and by Kalai [10] showing that the fractional Helly theorem holds with $\beta(\alpha, d) = 1 - (1-\alpha)^{1/(d+1)}$; yet another simplified proof of this fact has been subsequently given by Alon and Kalai [2]. It is well known that this bound is sharp by considering a family $\mathcal{F}$ consisting of $\approx (1-(1-\alpha)^{1/(d+1)})n$ copies of $\mathbb{R}^d$ and $\approx (1-\alpha)^{1/(d+1)}n$ hyperplanes in general position; see, e.g., the introduction of [10].

The colorful Helly theorem of Lovász covers the case where the sets are colored by $d+1$ colors and only the "colorful" $(d+1)$-tuples of sets in $\mathcal{F}$ are considered. Given families $\mathcal{F}_1, \ldots, \mathcal{F}_{d+1}$ of sets in $\mathbb{R}^d$ a family of sets $\{F_1, \ldots, F_{d+1}\}$ is a *colorful $(d+1)$-tuple* if $F_i \in \mathcal{F}_i$ for $i \in [d+1]$, where $[n] := \{1, \ldots, n\}$ for a non-negative integer $n \geq 1$. (The reader may think of $\mathcal{F}$ from preceding theorems decomposed into color classes $\mathcal{F}_1, \ldots, \mathcal{F}_{d+1}$.)

▶ **Theorem 3** (The colorful Helly theorem [14, 4]). *Let $\mathcal{F}_1, \ldots, \mathcal{F}_{d+1}$ be finite nonempty families of convex sets in $\mathbb{R}^d$. Assume that every colorful $(d+1)$-tuple has a nonempty intersection. Then one of the families $\mathcal{F}_1, \ldots, \mathcal{F}_{d+1}$ has a nonempty intersection.*

Both the colorful Helly theorem and the fractional Helly theorem with optimal bounds imply the Helly theorem. The colorful one by setting $\mathcal{F}_1 = \cdots = \mathcal{F}_{d+1} = \mathcal{F}$ and the fractional one by setting $\alpha = 1$ giving $\beta(1, d) = 1$.

The preceding two theorems can be merged into the following colorful fractional Helly theorem:

▶ **Theorem 4** (The colorful fractional Helly theorem [5]). *For every $\alpha \in (0,1]$ and every non-negative integer d, there is $\beta_{\mathrm{col}} = \beta_{\mathrm{col}}(\alpha, d) \in (0,1]$ with the following property. Let $\mathcal{F}_1, \ldots, \mathcal{F}_{d+1}$ be finite nonempty families of convex sets in $\mathbb{R}^d$ of sizes $n_1, \ldots, n_{d+1}$, respectively. If at least $\alpha n_1 \cdots n_{d+1}$ of the colorful $(d+1)$-tuples have a nonempty intersection, then there is an $i \in [d+1]$ such that $\mathcal{F}_i$ contains a subfamily of size at least $\beta_{\mathrm{col}} n_i$ with a nonempty intersection.*

Bárány et al. proved the colorful fractional Helly theorem with the value $\beta_{\mathrm{col}}(\alpha, d) = \frac{\alpha}{d+1}$ and they used it as a lemma [5, Lemma 3] in a proof of a colorful variant of a (p, q)-theorem. Despite this, the optimal bound for β_{col} seems to be of independent interest. In particular, the bound on β_{col} has been subsequently improved by Kim [13] who showed that the colorful fractional Helly theorem is true with $\beta_{\mathrm{col}}(\alpha, d) = \max\{\frac{\alpha}{d+1}, 1-(d+1)(1-\alpha)^{1/(d+1)}\}$. On the other hand, the value of $\beta_{\mathrm{col}}(\alpha, d)$ cannot go beyond $1-(1-\alpha)^{1/(d+1)}$ because essentially

the same example as for the standard fractional Helly theorem applies in this setting as well – it is sufficient to set $n_1 = n_2 = \cdots = n_{d+1}$ and take $\approx (1-(1-\alpha)^{1/(d+1)})n_i$ copies of $\mathbb{R}^d$ and $\approx (1-\alpha)^{1/(d+1)}n_i$ hyperplanes in general position in each color class.[1] (Kim [13] provides a slightly different upper bound example showing the same bound.)

Coming back to the lower bound on $\beta_{\mathrm{col}}(\alpha, d)$, Kim explicitly conjectured that $1-(1-\alpha)^{1/(d+1)}$ is also a lower bound, thereby an optimal bound for the colorful fractional Helly theorem. He also provides a more refined conjecture, that we discuss slightly later (see Conjecture 8), which implies this lower bound. We prove the refined conjecture, and therefore the optimal bounds for the colorful fractional Helly theorem.

▶ **Theorem 5** (The optimal colorful fractional Helly theorem)**.** *Let $\mathcal{F}_1, \ldots, \mathcal{F}_{d+1}$ be finite nonempty families of convex sets in $\mathbb{R}^d$ of sizes $n_1, \ldots, n_{d+1}$, respectively. If at least $\alpha n_1 \cdots n_{d+1}$ of the colorful $(d+1)$-tuples have a nonempty intersection, for $\alpha \in (0,1]$, then there is an $i \in [d+1]$ such that $\mathcal{F}_i$ contains a subfamily of size at least $(1-(1-\alpha)^{1/(d+1)})n_i$ with a nonempty intersection.*

In the proof we follow the exterior algebra approach which has been used by Kalai [10] in order to provide optimal bounds for the standard fractional Helly theorem. We have to upgrade Kalai's proof to the colorful setting. This requires guessing the right generalization of several steps in Kalai's proof (in particular guessing the statement of Theorem 10 below). However, we honestly admit that after making these "guesses" we follow Kalai's proof quite straightforwardly.

Let us also compare one aspect of our proof with the previous proof of the weaker bound by Kim [13]: Kim's proof uses the colorful Helly theorem as a blackbox while our proof includes the proof of the colorful Helly theorem.

Last but not least, the exterior algebra approach actually allows to generalize Theorem 5 in several different directions. The extension to so called d-collapsible complexes is essentially mandatory for the well working proof while the other generalizations that we will present just follow from the method. We will discuss this in detail in forthcoming subsections of the introduction.

1.1 d-representable and d-collapsible complexes

The nerve and d-representable complexes. The important information in Theorems 1, 2, 3, 4, and 5 is which subfamilies have a nonempty intersection. This information can be stored in a simplicial complex called the nerve.

A *(finite abstract) simplicial complex* is a set system K on a finite *set of vertices* N such that whenever $A \in \mathsf{K}$ and $B \subseteq A$, then $B \in \mathsf{K}$. (The standard notation for the vertex set would be V but this notation will be more useful later on when we will often use capital letters such as R for some set and the corresponding lower case letters such as r for its size.) The elements of K are *faces* (a.k.a. *simplices*) of K. The *dimension* of a face $A \in \mathsf{K}$ is defined as $\dim A = |A| - 1$; this corresponds to representing A as an $(|A|-1)$-dimensional simplex. The dimension of K, denoted $\dim \mathsf{K}$, is the maximum of the dimensions of faces in K. A face of dimension k is a *k-face* in short. *Vertices* of K are usually identified with 0-faces, that is, $v \in N$ is identified with $\{v\} \in \mathsf{K}$. (Though the definition of simplicial complex allows that $\{v\} \notin \mathsf{K}$ for $v \in N$, in our applications we will always have $\{v\} \in \mathsf{K}$ for $v \in N$.) Given a

[1] At the end of Section 3 we discuss this example in full detail in more general context. However, in this special case, it is perhaps much easier to check directly that β_{col} cannot be improved due to this example.

family of sets $\mathcal{F}$, the *nerve* of $\mathcal{F}$ is the simplicial complex whose vertex set is $\mathcal{F}$ and whose faces are subfamilies with a nonempty intersection. A simplicial complex is *d-representable* if it is the nerve of a finite family of convex sets in $\mathbb{R}^d$.

As a preparation for the d-collapsible setting, we now restate Theorem 5 in terms of d-representable complexes. For this we need two more notions. Given a simplicial complex K and a subset U of the vertex set N, the *induced subcomplex* $\mathsf{K}[U]$ is defined as $\mathsf{K}[U] := \{A \in \mathsf{K} \colon A \subseteq U\}$. Now, let us assume that the vertex set N is split into $d+1$ pairwise disjoint subsets $N = N_1 \sqcup \cdots \sqcup N_{d+1}$ (we can think of this partition as coloring each vertex of N with one of the $d+1$ possible colors).[2] Then a *colorful d-face* is a d-face A, such that $|A \cap N_i| = 1$ for every $i \in [d+1]$.

▶ **Theorem 6** (Theorem 5 reformulated)**.** *Let K be a d-representable simplicial complex with the set of vertices $N = N_1 \sqcup \cdots \sqcup N_{d+1}$ divided into $d+1$ disjoint subsets. Let $n_i := |N_i|$ for $i \in [d+1]$ and assume that K contains at least $\alpha n_1 \cdots n_{d+1}$ colorful d-faces for some $\alpha \in (0,1]$. Then there is an $i \in [d+1]$ such that $\dim \mathsf{K}[N_i] \geq (1-(1-\alpha)^{1/(d+1)})n_i - 1$.*

Theorem 6 is indeed just a reformulation of Theorem 5: Considering $\mathcal{F}$ as disjoint union[3] $\mathcal{F} = \mathcal{F}_1 \sqcup \cdots \sqcup \mathcal{F}_{d+1}$, then K corresponds to the nerve of $\mathcal{F}$, colorful d-faces correspond to colorful $(d+1)$-tuples with nonempty intersection and the dimension of $\mathsf{K}[N_i]$ corresponds to the size of largest subfamily of $\mathcal{F}_i$ with nonempty intersection minus 1. (The shift by minus 1 between size of a face and dimension of a face is a bit unpleasant; however, we want to follow the standard terminology.)

***d*-collapsible complexes.** In [17] Wegner introduced an important class of simplicial complexes, so called d-collapsible complexes. They include all d-representable complexes, which is the main result of [17], while they admit quite simple combinatorial description which is useful for induction.

Given a simplicial complex K, we say that a simplicial complex K' arises from K by an *elementary d-collapse*, if there are faces $L, M \in \mathsf{K}$ with the following properties: (i) $\dim L \leq d-1$; (ii) M is the unique inclusion-wise maximal face which contains L; and (iii) $\mathsf{K}' = \mathsf{K} \setminus \{A \in \mathsf{K} \colon L \subseteq A\}$. A simplicial complex K is *d-collapsible* if there is a sequence of simplicial complexes $\mathsf{K}_0, \ldots, \mathsf{K}_\ell$ such that $\mathsf{K} = \mathsf{K}_0$; K_i arises from K_{i-1} by an elementary d-collapse for $i \in [\ell]$; and K_ℓ is the empty complex.

We will prove the following generalization of Theorem 6 (equivalently of Theorem 5).

▶ **Theorem 7** (The optimal colorful fractional Helly theorem for d-collapsible complexes)**.** *Let K be a d-collapsible simplicial complex with the set of vertices $N = N_1 \sqcup \cdots \sqcup N_{d+1}$ divided into $d+1$ disjoint subsets. Let $n_i := |N_i|$ for $i \in [d+1]$ and assume that K contains at least $\alpha n_1 \cdots n_{d+1}$ colorful d-faces for some $\alpha \in (0,1]$. Then there is an $i \in [d+1]$ such that $\dim \mathsf{K}[N_i] \geq (1-(1-\alpha)^{1/(d+1)})n_i - 1$.*

1.2 Kim's refined conjecture and further generalization

As a tool for a possible proof of Theorem 5, Kim [13, Conjecture 4.2] suggested the following conjecture. (The notation k_i in Kim's statement of the conjecture is our $r_i + 1$.)

---

[2] We use the notation $\sqcup$ to emphasize the disjoint union.

[3] If there are any repetitions of sets in $\mathcal{F}$, which we generally allow for families of sets, then each repetition creates a new vertex in the nerve.

▶ **Conjecture 8** ([13]). *Let n_i be positive and r_i non-negative integers for $i \in [d+1]$ with $n_i \geq r_i + 1$. Let $\mathcal{F}_1, \ldots, \mathcal{F}_{d+1}$ be families of convex sets in $\mathbb{R}^d$ such that $|\mathcal{F}_i| = n_i$ and there is no subfamily of $\mathcal{F}_i$ of size $r_i + 1$ with non-empty intersection for every $i \in [d+1]$. Then the number of colorful $(d+1)$-tuples with nonempty intersection is at most*

$$n_1 \cdots n_{d+1} - (n_1 - r_1) \cdots (n_{d+1} - r_{d+1}).$$

We explicitly prove this conjecture in a slightly more general setting for d-collapsible complexes. (Note that the condition "no subfamily of size $r_i + 1$" translates as "no r_i-face", that is, "the dimension is at most $r_i - 1$".)

▶ **Proposition 9.** *Let n_i be positive and r_i non-negative integers for $i \in [d+1]$ with $n_i \geq r_i + 1$. Let K be a d-collapsible simplicial complex with the set of vertices $N = N_1 \sqcup \cdots \sqcup N_{d+1}$ divided into $d+1$ disjoint subsets such that $|N_i| = n_i$. Assume that $\dim \mathsf{K}[N_i] \leq r_i - 1$ for every $i \in [d+1]$. Then K contains at most*

$$n_1 \cdots n_{d+1} - (n_1 - r_1) \cdots (n_{d+1} - r_{d+1})$$

colorful d-faces.

Our main technical result. Now, let us present our main technical tool for a proof of Proposition 9 and consequently for a proof of Theorem 7 as well.

We denote by $\mathbb{N}$ the set of positive integers whereas $\mathbb{N}_0$ is the set of non-negative integers. Let us consider $c \in \mathbb{N}$ and vectors $\mathbf{k} = (k_1, \ldots, k_c), \mathbf{r} = (r_1, \ldots, r_c) \in \mathbb{N}_0^c$ and $\mathbf{n} = (n_1, \ldots, n_c) \in \mathbb{N}^c$ such that $\mathbf{k}, \mathbf{r} \leq \mathbf{n}$. (Here the notation $\mathbf{a} \leq \mathbf{b}$ means that $\mathbf{a}$ is less or equal to $\mathbf{b}$ in every coordinate.) We will also use the notation $k := k_1 + \cdots + k_c$, $n := n_1 + \cdots + n_c$, and $r := r_1 + \cdots + r_c$. Let N be a set with n elements partitioned as $N = N_1 \sqcup \cdots \sqcup N_c$ where $|N_i| = n_i$ for $i \in [c]$. By $\binom{N}{\mathbf{k}}$ we denote the set of all subsets A of N such that $|A \cap N_i| = k_i$ for every $i \in [c]$. Note that $\binom{N}{\mathbf{k}} \subseteq \binom{N}{k}$ where $\binom{N}{k}$ denotes the set of all subsets of N of size k.

Let K be a simplicial complex with the vertex set N as above. We say that a face A of K is $\mathbf{k}$-colorful if $A \in \binom{N}{\mathbf{k}}$, that is, $|A \cap N_i| = k_i$ for every $i \in [c]$. The earlier notion of colorful face corresponds to setting $c = d+1$ and $\mathbf{k} = \mathbf{1} := (1, \ldots, 1) \in \mathbb{N}^c$. By $f_{\mathbf{k}} = f_{\mathbf{k}}(\mathsf{K})$ we denote the $\mathbf{k}$-*colorful f-vector* of K, that is, the number of $\mathbf{k}$-colorful faces in K.

Let us further assume that we are given sets $R_i \subseteq N_i$ with $|R_i| = r_i$ for every $i \in [c]$. Let $R = R_1 \sqcup \cdots \sqcup R_c$ and $\bar{R} := N \setminus R$. Then, for $\mathbf{n}, \mathbf{r}$ as above and a positive integer d, we define the set system

$$P_{\mathbf{k}}(\mathbf{n}, d, \mathbf{r}) = \left\{ S \in \binom{N}{\mathbf{k}} : |S \cap \bar{R}| \leq d \right\}.$$

We remark that $P_{\mathbf{k}}(\mathbf{n}, d, \mathbf{r})$ is not a simplicial complex, as it contains only sets in $\binom{N}{\mathbf{k}}$. However, this set system is useful for estimating the number of $\mathbf{k}$-colorful faces in a d-collapsible complex.

By $p_{\mathbf{k}}(\mathbf{n}, d, \mathbf{r})$ we denote the size of $P_{\mathbf{k}}(\mathbf{n}, d, \mathbf{r})$, that is, $p_{\mathbf{k}}(\mathbf{n}, d, \mathbf{r}) := |P_{\mathbf{k}}(\mathbf{n}, d, \mathbf{r})|$.

▶ **Theorem 10.** *For integers $c, d \geq 1$, let K be a d-collapsible simplicial complex with vertex partition $N = N_1 \sqcup \cdots \sqcup N_c$ and let $\mathbf{n} = (n_1, \ldots, n_c) \in \mathbb{N}^c$ be the vector with $n_i = |N_i|$. For $\mathbf{r} = (r_1, \ldots, r_c) \in \mathbb{N}^c$ such that $\dim \mathsf{K}[N_i] \leq r_i - 1$ for $i \in [c]$ and $\mathbf{k} \in \mathbb{N}_0^c$ such that $\mathbf{k} \leq \mathbf{n}$ it follows that*

$$f_{\mathbf{k}}(\mathsf{K}) \leq p_{\mathbf{k}}(\mathbf{n}, d, \mathbf{r}).$$

$$\begin{array}{ccccc} \text{Thm. 10} & \Rightarrow & \text{Prop. 9} & \Rightarrow & \text{Thm. 7} \\ & & \Downarrow & & \Downarrow \\ & & \text{Conj. 8} & \Rightarrow & \text{Thm. 6} \Leftrightarrow \text{Thm. 5} \end{array}$$

Figure 1 The diagram of implications from Theorem 10 to Theorem 5. The top implications are proved below the statement of Theorem 10. The vertical implications follow from the main result of [17]. The equivalence at the bottom line has been explained below the statement of Theorem 6. Finally, note that we do not really need the implication Conj. 8 ⇒ Thm. 6. It comes from [13] where it also appears without explicit proof but it is easy – the interested reader may reconstruct it by checking the implication Prop. 9 ⇒ Thm. 7.

Theorem 10 is proved in Section 2. Here we show the implications Theorem 10 ⇒ Proposition 9 and Proposition 9 ⇒ Theorem 7. In addition, we advertise that Theorem 10 yields further generalizations of Theorem 7. We explain this last part in Section 3. For the convenience of the reader, we survey all the implications between the six preceding statements in Figure 1.

Proof of Proposition 9 modulo Theorem 10. We use Theorem 10 with $c = d+1$ and $\mathbf{k} = \mathbf{1}$. Then it is sufficient to compute $p_{\mathbf{1}}(\mathbf{n}, d, \mathbf{r})$. On the one hand, the size of $\binom{N}{\mathbf{1}}$ is $n_1 \dots n_{d+1}$. On the other hand, A belongs to $\binom{N}{\mathbf{1}} \setminus P_{\mathbf{1}}(\mathbf{n}, d, \mathbf{r})$ if and only if $|A \cap (N_i \setminus R_i)| = 1$ for every $i \in [d+1]$. Then, the number of such A is $(n_1 - r_1) \cdots (n_{d+1} - r_{d+1})$. Combining these observations we obtain the required formula

$$p_{\mathbf{1}}(\mathbf{n}, d, \mathbf{r}) = n_1 \dots n_{d+1} - (n_1 - r_1) \cdots (n_{d+1} - r_{d+1}). \qquad \blacktriangleleft$$

Proof of Theorem 7 modulo Proposition 9. By contradiction, let us assume that for every $i \in [d+1]$ we get $\dim \mathsf{K}[N_i] < (1 - (1-\alpha)^{1/(d+1)})n_i - 1$. Let us set $r_i := \dim \mathsf{K}[N_i] + 1 < (1 - (1-\alpha)^{1/(d+1)})n_i$. Then Proposition 9 gives that the number of colorful d-faces is at most

$$\prod_{i=1}^{d+1} n_i - \prod_{i=1}^{d+1} (n_i - r_i) < \prod_{i=1}^{d+1} n_i - (1 - (1 - (1-\alpha)^{1/(d+1)}))^{d+1} \prod_{i=1}^{d+1} n_i = \alpha \prod_{i=1}^{d+1} n_i$$

which is a contradiction due to the strict inequality on the line above. $\blacktriangleleft$

A topological version? A simplicial complex K is *d-Leray* if the ith reduced homology group $\tilde{H}_i(\mathsf{L})$ (over $\mathbb{Q}$) vanishes for every induced subcomplex $\mathsf{L} \leq \mathsf{K}$ and every $i \geq d$. As we already know, every d-representable complex is d-collapsible, and in addition every d-collapsible complex is d-Leray [17]. Helly-type theorems usually extend to d-Leray complexes and such extensions are interesting because they allow topological versions of Helly-type when collections of convex sets are replaced with good covers. We refer to several concrete examples [9, 11, 3] or to the survey [16].

We conjecture that it should be possible to extend Theorem 7 to d-Leray complexes and probably Theorem 10 as well. In the full version [6], we briefly discuss a possible approach but also a difficulty in that approach.

▶ **Conjecture 11** (The optimal colorful fractional Helly theorem for d-Leray complexes)**.** *Let K be a d-Leray simplicial complex with the set of vertices $N = N_1 \sqcup \dots \sqcup N_{d+1}$ divided into $d+1$ disjoint subsets. Let $n_i := |N_i|$ for $i \in [d+1]$ and assume that K contains at least $\alpha n_1 \cdots n_{d+1}$ colorful d-faces for some $\alpha \in (0, 1]$. Then there is an $i \in [d+1]$ such that* $\dim \mathsf{K}[N_i] \geq (1 - (1-\alpha)^{1/(d+1)})n_i - 1$.

2 Exterior algebra

In this section we prove Theorem 10. First we overview the required tools from exterior algebra – here we follow [10, Section 2] very closely.

Let N be a finite set of n elements (with a fixed total order $\leq$) and let $V = \mathbb{R}^N$ be the n-dimensional real vector space with standard basis vectors e_i for $i \in N$. Let $\bigwedge V$ be the 2^n dimensional exterior algebra over V with basis vectors e_S for $S \subseteq N$. The exterior product $\wedge$ on this algebra is defined so that it satisfies (i) $e_\emptyset$ is a neutral element, that is $e_\emptyset \wedge e_S = e_S = e_S \wedge e_\emptyset$; (ii) $e_S = e_{i_1} \wedge \cdots \wedge e_{i_s}$ for $S = \{i_1, \ldots, i_s\} \subseteq N$ where $i_1 < \cdots < i_s$ and we identify e_i with $e_{\{i\}}$ for $i \in N$ (iii) $e_i \wedge e_j = -e_j \wedge e_i$ for $i, j \in N$. By $\bigwedge^k V$ we denote the subspace of $\bigwedge V$ generated by $(e_S)_{S \in \binom{N}{k}}$ where $0 \leq k \leq n$. We consider the standard inner product on both V and $\bigwedge V$ so that $(e_i)_{i \in N}$ and $(e_S)_{S \subseteq N}$ are their orthonormal bases, respectively. Then $(e_S)_{S \in \binom{N}{k}}$ is also an orthonormal basis of $\bigwedge^k V$.

Given another basis $(g_i)_{i \in N}$, let $A = (a_{ij})_{i,j \in N}$ be the $N \times N$ transition matrix[4] from $(e_i)_{i \in N}$ to $(g_i)_{i \in N}$, that is, $g_i = \sum_{j \in N} a_{ij} e_j$ for any $i \in N$. The basis $(g_i)_{i \in N}$ induces a basis of $\bigwedge V$ given by $g_S = g_{i_1} \wedge \cdots \wedge g_{i_s}$ for $S = \{i_1, \ldots, i_s\} \subseteq N$. Transition from the standard basis $(e_S)_{S \in \binom{N}{k}}$ of $\bigwedge^k V$ to $(g_S)_{S \in \binom{N}{k}}$ is given by

$$g_S = \sum_{T \in \binom{N}{k}} \det(A_{S|T}) e_T \tag{1}$$

where $A_{S|T} = (a_{ij})_{i \in S, j \in T}$ for $S, T \subseteq N$.

Given an m-element set M and $M \times N$-matrix A and $k \leq m, n$, let $C_k(A)$ be the *compound* matrix $(\det A_{S|T})_{S \in \binom{M}{k}, T \in \binom{N}{k}}$.

The following lemma is implicitly contained in [10].

▶ **Lemma 12.** *If the columns of A are linearly independent, then the columns of $C_k(A)$ are linearly independent as well.*

Proof. If columns of A are linearly independent, then $n \leq m$. Consider an arbitrary square submatrix B of rank n. Considering B as a transition matrix from $(e_i)_{i \in N}$ to $(g_i)_{i \in N}$, we get that $C_k(B)$ is a transition matrix from $(e_S)_{S \in \binom{N}{k}}$ to $(g_S)_{S \in \binom{N}{k}}$, thus $C_k(B)$ has full rank. However, $C_k(B)$ is also a submatrix of $C_k(A)$ with all $\binom{n}{k}$ columns. ◀

Now, let us in addition assume that $(g_i)_{i \in N}$ is an orthonormal basis of V. As pointed out by Kalai, it follows from the Cauchy-Binet formula that $(g_S)_{S \in \binom{N}{k}}$ is also an orthonormal basis of $\bigwedge^k V$.

For $f, g \in \bigwedge V$ we define its *left interior product*, denoted by $g \llcorner f$, as the unique element in $\bigwedge V$ which satisfies that $\langle u, g \llcorner f \rangle = \langle u \wedge g, f \rangle$ for all $u \in \bigwedge V$. It turns out that $g_T \llcorner g_S$ is non-zero only if $T \subseteq S$, in which case $g_T \llcorner g_S = \pm g_{S \setminus T}$. (The sign is uniquely determined, but we do not need to express it explicitly.)

Colored exterior algebra. Now we extend the previous tools to the colored setting. From now on, let us assume that N is an n-element set decomposed into c-color classes, $N = N_1 \sqcup \cdots \sqcup N_c$. (The total order on N in this case starts with elements of N_1, then continues with elements

[4] Here we index rows and columns of a matrix by elements from some set, not necessarily integers. That is by $N \times N$ matrix we mean the matrix where both rows and columns are indexed by elements of N.

of N_2, etc.) We pick an $N \times N$-matrix A so that it is a block-diagonal matrix with blocks corresponding to individual N_i. That is, $A_{N_i|N_j}$ is a zero matrix whenever $i \neq j$. On the other hand, as shown by Kalai [10, Section 2], it is possible to pick each $A_{N_i|N_i}$ so that $(g_j)_{j\in N_i}$ is an orthonormal basis of the subspace of V generated by $(e_j)_{j\in N_i}$ and each square submatrix of $A_{N_i|N_i}$ has full rank. Therefore, from now on, we assume that we picked A and the vectors g_j this way. (Such a block matrix, for $c = 2$, is previously mentioned in [15].)

Similarly as in the introduction, let us set $\mathbf{n} = (n_1, \dots, n_c)$ so that $n_i = |N_i|$ for $i \in [c]$; for simplicity, let us assume that each N_i is nonempty – that is, $\mathbf{n}$ is a c-tuple of positive integers. Let us also consider another c-tuple $\mathbf{k} = (k_1, \dots, k_c)$ of non-negative integers such that $\mathbf{k} \leq \mathbf{n}$ and we set $k := k_1 + \cdots + k_c$. Then by $\bigwedge^{\mathbf{k}} V$ we mean the subspace of $\bigwedge V$ generated by $(e_S)_{S\in\binom{N}{\mathbf{k}}}$; recall that $\binom{N}{\mathbf{k}}$ is the set of all subsets A of N such that $|A\cap N_i| = k_i$ and that $\binom{N}{\mathbf{k}} \subseteq \binom{N}{k}$. Thus we also get that $\bigwedge^{\mathbf{k}} V$ is a subspace of $\bigwedge^k V$. In addition, due to our choice of $(g_j)_{j\in N}$ we get that $g_S \in \bigwedge^{\mathbf{k}} V$ if $S \in \binom{N}{\mathbf{k}}$. In addition $\det A_{S|T} = 0$ if $T \in \binom{N}{k} \setminus \binom{N}{\mathbf{k}}$ because $A_{S|T}$ is in this case a block matrix such that some of the blocks is not a square. Thus the formula (1) simplifies to

$$g_S = \sum_{T\in\binom{N}{\mathbf{k}}} \det(A_{S|T}) e_T. \tag{2}$$

Proof of Theorem 10. For $\mathbf{k} \in \mathbb{N}^c$ such that $k \leq d$ we have that $P_{\mathbf{k}}(\mathbf{n}, d, \mathbf{r}) = \binom{N}{\mathbf{k}}$, thus the theorem follows trivially. On the other hand, if $k > r$, then $k_i > r_i$ for some i and consequently $f_{\mathbf{k}}(\mathsf{K}) = 0$ due to our assumption $\dim \mathsf{K}[N_i] \leq r_i - 1$; therefore the theorem again follows trivially. From now on we assume $d + 1 \leq k \leq r$. (We also use the notation for the sets R, $\bar{R}$ and R_i with $|R_i| = r_i$ as in the definition of $P_{\mathbf{k}}(\mathbf{n}, d, \mathbf{r})$.)

Let us define the following subspaces of $\bigwedge^{\mathbf{k}} V$

$$A_{\mathbf{k}} := \left\{ m \in \bigwedge^{\mathbf{k}} V : \left(\forall T \in \binom{R}{k-d}\right)\ g_T \llcorner m = 0 \right\},$$

and

$$W_{\mathbf{k}} := \operatorname{span}\left\{ e_S \in \bigwedge^{\mathbf{k}} V : S \in \binom{N}{\mathbf{k}} \text{ and } S \in \mathsf{K} \right\},$$

from the definition it follows that the colorful f-vector and the dimension of $W_{\mathbf{k}}$ coincide, i.e. $f_{\mathbf{k}} = \dim(W_{\mathbf{k}})$.

We claim that

$$\dim(A_{\mathbf{k}}) \geq \left|\binom{N}{\mathbf{k}}\right| - p_{\mathbf{k}}(\mathbf{n}, d, \mathbf{r}).$$

Indeed, if $S \in \binom{N}{\mathbf{k}}$ such that $S \notin P_{\mathbf{k}}(\mathbf{n}, d, \mathbf{r})$, then $|S \cap \bar{R}| > d$. As $S \subseteq R \sqcup \bar{R} = N$ and $|S| = k$ we have that $|S \cap R| < k - d$. If $T \in \binom{R}{k-d}$ we have that $S \not\supseteq T$; therefore $g_T \llcorner g_S = 0$. From this it follows that $g_S \in A_{\mathbf{k}}$ and finally the claim because $g_S \in \bigwedge^{\mathbf{k}} V$.

The core of the proof is to show $A_{\mathbf{k}} \cap W_{\mathbf{k}} = \{0\}$. Once we have this, we get $f_k(\mathsf{K}) = \dim(W_k) \leq \dim \bigwedge^{\mathbf{k}} V - \dim A_{\mathbf{k}} \leq p_{\mathbf{k}}(\mathbf{n}, d, \mathbf{r})$ which proves the theorem.

For contradiction, let $m \in A_k \cap W_k$ be a non-zero element. Because $m \in W_{\mathbf{k}}$, it can be written as $m = \sum \alpha_S e_S$ where the sum is over all $S \in \binom{N}{\mathbf{k}}$ such that $S \in \mathsf{K}$. Let $\mathsf{K}_0, \dots, \mathsf{K}_\ell$ be a sequence of simplicial complexes showing d-collapsibility of K. In addition, due to [10,

Lemma 3.2], it is possible to assume that K_i arises from K_{i-1} by so called *special* elementary d-collapse which is either a removal of a maximal face of dimension at most $d-1$ or the minimal face (the face L in the definition) has dimension exactly $d-1$.

Now let us consider the first step from K_{i-1} to K_i such that a face $U \in \binom{N}{\mathbf{k}}$ with non-zero α_U is eliminated. Denote by L and M the faces determining the collapse as in the definition. We have $L \subseteq U \subseteq M$, $|M| \geq |U| = k > d$ and therefore $|L| = d$ (equivalently, $\dim L = d-1$), because the collapse is special. For $T \in \binom{R}{k-d}$ let $\mathbf{t} = (t_1, \dots, t_c) \in \mathbb{N}^c$ be such that $t_i = |T \cap N_i|$. Then $g_T = \sum_{P \in \binom{N}{\mathbf{t}}} \det(A_{T|P}) e_P$ via (2). We also need to simplify the expression $\langle e_L, g_T \llcorner e_S \rangle$ for $S \in \binom{N}{\mathbf{k}}$. We obtain

$$\langle e_L, g_T \llcorner e_S \rangle = \langle e_L \wedge g_T, e_S \rangle = \sum_{P \in \binom{N}{\mathbf{t}}} \det(A_{T|P}) \langle e_L \wedge e_P, e_S \rangle \tag{3}$$

If $S \not\supseteq L$ then $\langle e_L \wedge e_P, e_S \rangle = 0$ for all P, and therefore $\langle e_L, g_T \llcorner e_S \rangle = 0$. If $S \supseteq L$ then $\langle e_L \wedge e_P, e_S \rangle = 0$ unless $P = S \setminus L$ and therefore $\langle e_L, g_T \llcorner e_S \rangle = \langle e_L \wedge e_{S \setminus L}, e_S \rangle \det(A_{T|S \setminus L})$.

Since $m \in A_k$, for arbitrary $T \in \binom{R}{k-d}$ we get

$$\begin{aligned} 0 = \langle e_L, g_T \llcorner m \rangle &= \sum_{S \in \binom{N}{\mathbf{k}} : S \in \mathsf{K}} \alpha_S \langle e_L, g_T \llcorner e_S \rangle = \sum_{S \in \binom{N}{\mathbf{k}} : S \in \mathsf{K}_{i-1}} \alpha_S \langle e_L, g_T \llcorner e_S \rangle \\ &= \sum_{S \in \binom{N}{\mathbf{k}} : S \supseteq L} \alpha_S \langle e_L, g_T \llcorner e_S \rangle = \sum_{S \in \binom{N}{\mathbf{k}} : M \supseteq S \supseteq L} \alpha_S \langle e_L \wedge e_{S \setminus L}, e_S \rangle \det(A_{T|S \setminus L}) \end{aligned}$$

where the third equality follows from the fact that $\alpha_S = 0$ for $S \in \mathsf{K} \setminus \mathsf{K}_{i-1}$ due to our choice of K_{i-1} and the last two equalities follow from our earlier simplification of $\langle e_L, g_T \llcorner e_S \rangle$. (We also use that the expressions $S \supseteq L$ and $M \supseteq S \supseteq L$ are equivalent as M is the unique maximal face containing L.)

We also have $U \in \binom{N}{\mathbf{k}}$ with $M \supseteq U \supseteq L$ for which $\alpha_U \neq 0$ as well as $\langle e_L \wedge e_{U \setminus L}, e_U \rangle$ is nonzero (the latter one equals ± 1). Therefore the expression above is a linear dependence of the columns of $C_{k-d}(A_{R|M \setminus L})$. However, we will also show that the columns of $C_{k-d}(A_{R|M \setminus L})$ are linearly independent, thereby getting a contradiction. Via Lemma 12, it is sufficient to check that the columns of $A_{R|M \setminus L}$ are linearly independent. Because A is a block-matrix with blocks $A_{N_i|N_i}$, we get that $A_{R|M \setminus L}$ is a block matrix with blocks $A_{R_i|(M \setminus L) \cap N_i}$. Thus it is sufficient to check that the columns are independent in each block. But this follows from our assumptions of how we picked A in each block, using that $|R_i| = r_i \geq |(M \setminus L) \cap N_i|$ as $|M \cap N_i| \leq r_i$ due to our assumption $\dim \mathsf{K}[V_i] \leq r_i - 1$. ◀

3 k-colorful fractional Helly theorem

Theorem 10 allows to generalize Theorem 7 in two more directions.

The first generalization of Theorem 7 is already touched in the introduction. We can deduce an analogy of Theorem 7 for $\mathbf{k}$-colorful faces (instead of just colorful d-faces) where $\mathbf{k} = (k_1, \dots, k_c) \in \mathbb{N}_0^c$ is some vector with $c \geq 1$. For example, if $d = 2$, $\mathbf{k} = (2, 1, 1)$ and we understand the partition of $N = N_1 \sqcup N_2 \sqcup N_3$ as coloring the vertices of K red, green, or blue. Then we seek for the number of faces that contain two red vertices, one green vertex and one blue vertex.

For the second generalization, let us first observe that in the conclusion of Theorem 7 there is the same coefficient $1 - (1-\alpha)^{1/(d+1)}$ independently of i. However, in the notation of Theorem 7, we may also seek for i such that $\dim \mathsf{K}[N_i] \geq \beta_i n_i + 1$ where $\boldsymbol{\beta} = (\beta_1, \dots, \beta_c) \in$

$(0,1]^c$ is some fixed vector. Then for given $\boldsymbol{\beta}$, we want to find the lowest $\alpha \in (0,1]$ with which we reach the conclusion analogous as in Theorem 7. This is a natural analogy of various Ramsey type statements: for example, if the edges of a complete graph G with at least 9 vertices are colored blue or red, then the graph contains either a blue copy of the complete graph on 3 vertices or a red copy of the complete graph on 4 vertices.

For the purpose of stating the generalization, let us set

$$L_{\mathbf{k}}(d) := \{\boldsymbol{\ell} = (\ell_1, \cdots \ell_c) \in \mathbb{N}_0^c \colon \ell_1 + \cdots + \ell_c \leq d \text{ and } \ell_i \leq k_i \text{ for } i \in [c]\} \tag{4}$$

and

$$\alpha_{\mathbf{k}}(d, \boldsymbol{\beta}) := \sum_{\boldsymbol{\ell}=(\ell_1,\ldots,\ell_c)\in L_{\mathbf{k}}(d)} \prod_{i=1}^{c} \binom{k_i}{\ell_i} (1-\beta_i)^{\ell_i} (\beta_i)^{k_i-\ell_i}. \tag{5}$$

▶ **Theorem 13.** *Let $c, d \geq 1$ and $\mathbf{k} = (k_1, \ldots, k_c) \in \mathbb{N}_0^c$ be such that $k := k_1 + \cdots + k_c \geq d+1$. Let K be a d-collapsible simplicial complex with the set of vertices $N = N_1 \sqcup \cdots \sqcup N_c$ divided into c disjoint subsets. Let $n_i := |N_i|$ for $i \in [c]$ and assume that K contains at least $\alpha_{\mathbf{k}}(d, \boldsymbol{\beta})\left|\binom{N}{\mathbf{k}}\right|$ $\mathbf{k}$-colorful faces for some $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_c) \in (0,1]^c$. Then there is an $i \in [c]$ such that $\dim \mathsf{K}[N_i] \geq \beta_i n_i - 1$.*

The formula (5) for $\alpha_{\mathbf{k}}(d, \boldsymbol{\beta})$ in Theorem 13 is, unfortunately, a bit complicated. However, this is the optimal value for α in the theorem. We first prove Theorem 13 and then we will provide an example showing that for every d, $\mathbf{k}$ and $\boldsymbol{\beta}$ as in the theorem, the value for α cannot be improved. The remark below is a probabilistic interpretation of (5). (This, for example, easily reveals that $\alpha_{\mathbf{k}}(d, \boldsymbol{\beta}) \in (0,1]$ for given parameters and will help us with checking monotonicity in $\boldsymbol{\beta}$.)

▶ Remark 14. Consider a random experiment where we gradually for each i pick k_i numbers $x_1^i, \ldots, x_{k_i}^i$ in the interval $[0,1]$ independently at random (with uniform distribution). Let ℓ_i be the number of x_j^i which are greater than β_i and let us consider the event $A_{\mathbf{k}}(d, \boldsymbol{\beta})$ expressing that $\ell_1 + \cdots + \ell_c \leq d$. Then $\alpha_{\mathbf{k}}(d, \boldsymbol{\beta})$ is the probability $\mathbf{P}[A_{\mathbf{k}}(d, \boldsymbol{\beta})]$. Indeed, the probability that the number of x_j^i which are greater than β_i is exactly ℓ_i is given by the expression beyond the sum in (5). Therefore, we need to sum this over all options giving $\ell_1 + \cdots + \ell_c \leq d$ and $\ell_i \leq k_i$.

In the proof of Theorem 13 we will need the following slightly modified proposition. We relax "at least" to "more than" while we aim at a strict inequality in the conclusion – this innocent change will be a significant advantage in the proof. On the other hand, after this change we can drop the assumption $k \geq d+1$. But this is only a cosmetic change, because the proposition below is vacuous if $\alpha_{\mathbf{k}}(d, \boldsymbol{\beta}) = 1$ which in particular happens if $k < d+1$.

▶ **Proposition 15.** *Let $c, d \geq 1$ and $\mathbf{k} = (k_1, \ldots, k_c) \in \mathbb{N}_0^c$. Let K be a d-collapsible simplicial complex with the set of vertices $N = N_1 \sqcup \cdots \sqcup N_c$ divided into c disjoint subsets. Let $n_i := |N_i|$ for $i \in [c]$ and assume that K contains more than $\alpha_{\mathbf{k}}(d, \boldsymbol{\beta})\left|\binom{N}{\mathbf{k}}\right|$ $\mathbf{k}$-colorful faces for some $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_c) \in (0,1]^c$. Then there is an $i \in [c]$ such that $\dim \mathsf{K}[N_i] > \beta_i n_i - 1$.*

First we show how Theorem 13 follows from Proposition 15 by a limit transition. Then we prove Proposition 15.

Proof of Theorem 13 modulo Proposition 15. Let us consider $\varepsilon > 0$ such that $\boldsymbol{\beta} - \boldsymbol{\varepsilon} \in (0,1]^c$ for $\boldsymbol{\varepsilon} = (\varepsilon, \ldots, \varepsilon) \in (0,1]^c$.

First, we need to check $\alpha_{\mathbf{k}}(d, \boldsymbol{\beta}) > \alpha_{\mathbf{k}}(d, \boldsymbol{\beta} - \boldsymbol{\varepsilon})$. For this we will use Remark 14 and we also use $k \geq d+1$. It is easy to check $A_{\mathbf{k}}(d, \boldsymbol{\beta}) \supseteq A_{\mathbf{k}}(d, \boldsymbol{\beta} - \boldsymbol{\varepsilon})$ which gives $\alpha_{\mathbf{k}}(d, \boldsymbol{\beta}) \geq \alpha_{\mathbf{k}}(d, \boldsymbol{\beta} - \boldsymbol{\varepsilon})$. In order to show the strict inequality, it remains to show that $A_{\mathbf{k}}(d, \boldsymbol{\beta}) \setminus A_{\mathbf{k}}(d, \boldsymbol{\beta} - \boldsymbol{\varepsilon})$ has

positive probability. Consider the output of the experiment when each $x_i^j \in (\beta_i - \varepsilon, \beta_i)$. This output has positive probability ε^k. In addition, this output belongs to $A_{\mathbf{k}}(d, \boldsymbol{\beta})$ whereas it does not belong to $A_{\mathbf{k}}(d, \boldsymbol{\beta} - \boldsymbol{\varepsilon})$ (because $k \geq d+1$) as required.

This means, that we can apply Proposition 15 with $\alpha_{\mathbf{k}}(d, \boldsymbol{\beta} - \boldsymbol{\varepsilon})$ as we know that K has at least $\alpha_{\mathbf{k}}(d, \boldsymbol{\beta})\left|\binom{N}{\mathbf{k}}\right|$ $\mathbf{k}$-colorful faces by assumptions of Theorem 13 which is more than $\alpha_{\mathbf{k}}(d, \boldsymbol{\beta} - \varepsilon)\left|\binom{N}{\mathbf{k}}\right|$. We obtain $\dim \mathsf{K}[N_i] > (\beta_i - \varepsilon)n_i - 1$. By letting ε tend to 0, we obtain the required $\dim \mathsf{K}[N_i] \geq \beta_i n_i - 1$. ◀

Boosting the complex. In the proof of Proposition 15, we will need the following procedure for boosting the complex. For a given complex K with vertex set $N = N_1 \sqcup \cdots \sqcup N_c$ partitioned as usual, and a non-negative integer m we define the complex $K_{\langle m \rangle}$ as a complex with the vertex set $N \times [m] = N_1 \times [m] \sqcup \cdots \sqcup N_c \times [m]$ whose maximal faces are of the form $S \times [m]$, where S is a maximal face of K. We will also use the notation $\delta_{\mathbf{k}}(\mathsf{K}) := f_{\mathbf{k}}(K)/\left|\binom{N}{\mathbf{k}}\right|$ for the density of $\mathbf{k}$-colorful faces of K. The item (ii) of the following lemma is the contents of [1, Proposition 2.1]; we thank an anonymous referee for pointing out this reference to us.

▶ **Lemma 16.** *Let K be a simplicial complex with vertex partition $N = N_1 \sqcup \cdots \sqcup N_c$ and $\mathbf{k} = (k_1, \ldots, k_c) \in \mathbb{N}_0^c$, then*

(i) *$\delta_{\mathbf{k}}(\mathsf{K}_{\langle m \rangle}) \geq \delta_{\mathbf{k}}(\mathsf{K})$; and*

(ii) *if K is d-collapsible, then $\mathsf{K}_{\langle m \rangle}$ is d-collapsible as well.*

Proof. We prove only (i) as (ii) is the contents of [1, Proposition 2.1].

If $\delta_{\mathbf{k}}(\mathsf{K}) = 0$ there is nothing to prove. Thus we may assume that $\delta_{\mathbf{k}}(\mathsf{K}) > 0$ (equivalently $f_{\mathbf{k}}(\mathsf{K}) > 0$) and consequently we have that $|N_i| \geq k_i$. Let us interpret $\delta_{\mathbf{k}}(\mathsf{K})$ as the probability that a random $\mathbf{k}$-tuple of vertices in N is a simplex of K, and we interpret $\delta_{\mathbf{k}}(\mathsf{K}_{\langle m \rangle})$ analogously. Let $\pi\colon N \times [m] \to N$ be the projection to the first coordinate. Now, let U be a $\mathbf{k}$-tuple of vertices in $N \times [m]$ taken uniformly at random. Considering the set $\pi(U) \subseteq N$, it need not be a $\mathbf{k}$-tuple (this happens exactly when two points in U have the same image under π) but it can be extended to a $\mathbf{k}$-tuple W using that $|N_i| \geq k_i$ for every i. Let W be an extension of $\pi(U)$ to a $\mathbf{k}$-tuple, taken uniformly at random among all possible choices. Because of the choices we made, W is in fact a $\mathbf{k}$-tuple of vertices in N taken uniformly at random. (Note that the choices done in each N_i or $N_i \times [m]$ are independent of each other.) Altogether, using $\mathbf{P}$ for probability, we get

$$\delta_{\mathbf{k}}(\mathsf{K}_{\langle m \rangle}) = \mathbf{P}[U \in \mathsf{K}_{\langle m \rangle}] = \mathbf{P}[\pi(U) \in \mathsf{K}] \geq \mathbf{P}[W \in \mathsf{K}] = \delta_{\mathbf{k}}(\mathsf{K}). \qquad ◀$$

Density of $P_{\mathbf{k}}(\mathbf{n}, d, \mathbf{r})$. Now, we will provide a formula for the density of $P_{\mathbf{k}}(\mathbf{n}, d, \mathbf{r})$. In the following computations we also set $\delta_{\mathbf{k}}(\mathbf{n}, d, \mathbf{r}) = p_{\mathbf{k}}(\mathbf{n}, d, \mathbf{r})/\left|\binom{N}{\mathbf{k}}\right|$ using the notation from the definition of $P_{\mathbf{k}}(\mathbf{n}, d, \mathbf{r})$. We get

$$\begin{aligned} p_{\mathbf{k}}(\mathbf{n}, d, \mathbf{r}) &= \left|\left\{S \in \binom{N}{\mathbf{k}} : |S \cap \bar{R}| \leq d\right\}\right| \\ &= \sum_{\boldsymbol{\ell}=(\ell_1,\ldots,\ell_c) \in L_{\mathbf{k}}(d)} \left|\left\{S \in \binom{N}{\mathbf{k}} : |S_i \cap \bar{R}_i| = l_i\right\}\right| \\ &= \sum_{\boldsymbol{\ell}=(\ell_1,\ldots,\ell_c) \in L_{\mathbf{k}}(d)} \prod_{i=1}^{c} \binom{n_i - r_i}{l_i}\binom{r_i}{k_i - l_i}. \end{aligned}$$

Then, using $(x)_m := x \cdot (x-1) \cdots (x-(m-1))$, the density is given by

$$\delta_{\mathbf{k}}(\mathbf{n}, d, \mathbf{r}) = \frac{p_{\mathbf{k}}(\mathbf{n}, d, \mathbf{r})}{\prod_{i=1}^{c} \binom{n_i}{k_i}} = \frac{\sum_{\boldsymbol{\ell}=(\ell_1,\dots,\ell_c)\in L_{\mathbf{k}}(d)} \prod_{i=1}^{c} \binom{k_i}{\ell_i} (n_i - r_i)_{\ell_i} (r_i)_{k_i-\ell_i}}{\prod_{i=1}^{c} (n_i)_{k_i}}. \tag{6}$$

Proof of Proposition 15. For contradiction, let us assume that for every $i \in [c]$ we have that $\dim(\mathsf{K}[V_i]) \le \beta_i n_i - 1$. Let us set $r_i := \dim(\mathsf{K}[V_i]) + 1 \le \beta_i n_i$. Note that the conclusion of Theorem 10 can be restated as $\delta_{\mathbf{k}}(\mathsf{K}) \le \delta_{\mathbf{k}}(\mathbf{n}, d, \mathbf{r})$.

Now we get

$$\begin{aligned}
\delta_{\mathbf{k}}(\mathsf{K}) &\le \liminf_{m\to\infty} \delta_{\mathbf{k}}(\mathsf{K}_{\langle m\rangle}) \text{ by Lemma 16(i)} \\
&\le \liminf_{m\to\infty} \delta_{\mathbf{k}}(m\mathbf{n}, d, m\mathbf{r}) \text{ by Theorem 10 using Lemma 16(ii)} \\
&\le \liminf_{m\to\infty} \delta_{\mathbf{k}}(m\mathbf{n}, d, \lfloor mn_i\beta_i \rfloor) \text{ using } r_i \le \beta_i n_i \text{ and monotonicity of } p_{\mathbf{k}}(\mathbf{n}, d, \mathbf{r}) \text{ in } \mathbf{r} \\
&= \liminf_{m\to\infty} \frac{\sum_{\boldsymbol{\ell}=(\ell_1,\dots,\ell_c)\in L_{\mathbf{k}}(d)} \prod_{i=1}^{c} \binom{k_i}{\ell_i} (mn_i - \lfloor mn_i\beta_i\rfloor)_{\ell_i} (\lfloor mn_i\beta_i\rfloor)_{k_i-\ell_i}}{\prod_{i=1}^{c} (mn_i)_{k_i}} \text{ by (6)} \\
&= \sum_{\boldsymbol{\ell}=(\ell_1,\dots,\ell_c)\in L_{\mathbf{k}}(d)} \prod_{i=1}^{c} \binom{k_i}{\ell_i} (1-\beta_i)^{\ell_i} (\beta_i)^{k_i-\ell_i} \\
&= \alpha_{\mathbf{k}}(d, \boldsymbol{\beta})
\end{aligned}$$

which is a contradiction with the assumptions. ◀

▶ Remark 17. It would be much more natural to try to avoid boosting the complex and show directly $\delta_k(\mathsf{K}) \le \delta_k(\mathbf{n}, d, \mathbf{r}) \le \alpha_{\mathbf{k}}(d, \boldsymbol{\beta})$ in the proof of Proposition 15. The former inequality follows from Theorem 10. However, the latter inequality turned out to be somewhat problematic for us when we attempted to show it directly from the definition of $\alpha_{\mathbf{k}}(d, \boldsymbol{\beta})$ and from (6). Thus, in our computations, we take an advantage of the fact that the computations in the limit are easier.

Tightness of Theorem 13. We conclude this section by showing that the bound given in Theorem 13 is tight.

Let us fix $c, d \in \mathbb{N}$, $\mathbf{k} = (k_1, \dots, k_c) \in \mathbb{N}_0^c$ with $k := k_1 + \cdots + k_c \ge d+1$ and $\boldsymbol{\beta} = (\beta_1, \dots, \beta_c) \in (0,1]^c$ as in the statement of Theorem 13. Let $0 \le \alpha' < \alpha_{\mathbf{k}}(d, \boldsymbol{\beta})$. We will find a complex K which contains at least $\alpha' |\binom{N}{\mathbf{k}}|$ $\mathbf{k}$-colorful faces while $\dim \mathsf{K}[N_i] < \beta_i n_i - 1$ for every $i \in [c]$ (using the notation from the statement of Theorem 13).

Similarly as in the proof of Theorem 13 let us consider $\varepsilon > 0$ such that $\boldsymbol{\beta} - \boldsymbol{\varepsilon} \in (0,1]^c$ for $\boldsymbol{\varepsilon} = (\varepsilon, \dots, \varepsilon) \in (0,1]^c$. In addition, because $\alpha_{\mathbf{k}}(d, \boldsymbol{\beta})$ is continuous in $\boldsymbol{\beta}$ due to its definition (5), we may pick ε such that $\alpha' < \alpha_{\mathbf{k}}(d, \boldsymbol{\beta} - \boldsymbol{\varepsilon})$. For simplicity of notation, let $\boldsymbol{\beta}' = (\beta_1', \dots, \beta_c') := \boldsymbol{\beta} - \boldsymbol{\varepsilon}$.

Now we pick a positive integer m and set $\mathbf{n} = (m, \dots, m) \in \mathbb{N}^c$, that is, $n_1 = \cdots = n_c = m$ and $n = cm$ in our standard notation. We also set $\mathbf{r} = (r_1, \dots, r_c)$ so that $r_i := \lfloor \beta_i' m \rfloor$.[5] We assume that m is large enough so that $r_i \ge k_i$ for each $i \in [c]$. We define families N_i of convex sets in $\mathbb{R}^d$ so that each N_i contains r_i copies of $\mathbb{R}^d$ and $m - r_i$ hyperplanes in

[5] This choice of $\mathbf{n}$ will yield a counterexample where each color class has equal size. It would be also possible to vary the sizes.

general position. We also assume that the collection of all hyperplanes in $N_1, \dots, N_c$ is in general position. We set K to be the nerve of the family $N = N_1 \sqcup \dots \sqcup N_c$. In particular K is d-representable (therefore d-collapsible as well).

First, we check that $\dim \mathsf{K}[N_i] < \beta_i m - 1$ provided that m is large enough. A subfamily of N_i with nonempty intersection contains at most d hyperplanes from N_i. Therefore $\dim \mathsf{K}[N_i] < r_i + d = \lfloor \beta_i' m \rfloor + d < \beta_i m - 1$ for m large enough.

Next we check that K contains at least $\alpha' |\binom{N}{\mathbf{k}}|$ $\mathbf{k}$-colorful faces provided that m is large enough. Partitioning N_i so that R_i is the subfamily of the copies of $\mathbb{R}^d$ and $\bar{R}_i$ is the subfamily of hyperplanes, we get

$$f_{\mathbf{k}}(K) = p_{\mathbf{k}}(\mathbf{n}, d, \mathbf{r})$$

from the definition of $p_{\mathbf{k}}(\mathbf{n}, d, \mathbf{r})$. Therefore (6) gives

$$\delta_{\mathbf{k}}(\mathsf{K}) = \frac{\sum\limits_{\boldsymbol{\ell}=(\ell_1,\dots,\ell_c)\in L_{\mathbf{k}}(d)} \prod\limits_{i=1}^{c} \binom{k_i}{\ell_i} (m - \lfloor \beta_i' m \rfloor)_{\ell_i} (\lfloor \beta_i' m \rfloor)_{k_i - \ell_i}}{\prod_{i=1}^{c} (m)_{k_i}}.$$

Passing to the limit (considering the dependency of K on m), we get

$$\lim_{m\to\infty} \delta_{\mathbf{k}}(\mathsf{K}) = \sum_{\boldsymbol{\ell}=(\ell_1,\dots,\ell_c)\in L_{\mathbf{k}}(d)} \prod_{i=1}^{c} \binom{k_i}{\ell_i} (1-\beta_i')^{\ell_i} (\beta_i')^{k_i-\ell_i} = \alpha_{\mathbf{k}}(d, \boldsymbol{\beta}').$$

Therefore, for m large enough K contains at least $\alpha' |\binom{N}{\mathbf{k}}|$ $\mathbf{k}$-colorful as $\alpha' < \alpha_{\mathbf{k}}(d, \boldsymbol{\beta}')$.

References

1. R. Aharoni, R. Holzman, and Z. Jiang. Rainbow fractional matchings. *Combinatorica*, 39(6):1191–1202, 2019.
2. N. Alon and G. Kalai. A simple proof of the upper bound theorem. *European J. Combin.*, 6(3):211–214, 1985.
3. N. Alon, G. Kalai, J. Matoušek, and R. Meshulam. Transversal numbers for hypergraphs arising in geometry. *Adv. in Appl. Math.*, 29(1):79–101, 2002.
4. I. Bárány. A generalization of Carathéodory's theorem. *Discrete Math.*, 40(2-3):141–152, 1982.
5. I. Bárány, F. Fodor, L. Montejano, D. Oliveros, and A. Pór. Colourful and fractional (p, q)-theorems. *Discrete Comput. Geom.*, 51(3):628–642, 2014.
6. D. Bulavka, A. Goodarzi, and M. Tancer. Optimal bounds for the colorful fractional Helly theorem. Preprint, 2020. `arXiv:2010.15765`.
7. J. Eckhoff. An upper-bound theorem for families of convex sets. *Geom. Dedicata*, 19(2):217–227, 1985.
8. E. Helly. Über Mengen konvexer Körper mit gemeinschaftlichen Punkten. *Jahresber. Deustch. Math.-Verein.*, 32:175–176, 1923.
9. E. Helly. Über Systeme von abgeschlossenen Mengen mit gemeinschaftlichen Punkten. *Monaths. Math. und Physik*, 37:281–302, 1930.
10. G. Kalai. Intersection patterns of convex sets. *Israel J. Math.*, 48(2-3):161–174, 1984.
11. G. Kalai and R. Meshulam. A topological colorful Helly theorem. *Adv. Math.*, 191(2):305–311, 2005.
12. M. Katchalski and A. Liu. A problem of geometry in $\mathbf{R}^n$. *Proc. Amer. Math. Soc.*, 75(2):284–288, 1979.
13. M. Kim. A note on the colorful fractional Helly theorem. *Discrete Math.*, 340(1):3167–3170, 2017.
14. L. Lovász. Problem 206. *Matematikai Lapok*, 25:181, 1974.

15 E. Nevo. Algebraic shifting and basic constructions on simplicial complexes. *J. Algebraic Combin.*, 22(4):411–433, 2005.

16 M. Tancer. Intersection patterns of convex sets via simplicial complexes: A survey. In J. Pach, editor, *Thirty Essays on Geometric Graph Theory*, pages 521–540. Springer New York, 2013.

17 G. Wegner. d-collapsing and nerves of families of convex sets. *Arch. Math. (Basel)*, 26:317–321, 1975.

An Integer Programming Formulation Using Convex Polygons for the Convex Partition Problem

Hadrien Cambazard
Université Grenoble Alpes, CNRS, Grenoble INP, G-SCOP, 38000 Grenoble, France

Nicolas Catusse
Université Grenoble Alpes, CNRS, Grenoble INP, G-SCOP, 38000 Grenoble, France

Abstract

A convex partition of a point set P in the plane is a planar partition of the convex hull of P into empty convex polygons or internal faces whose extreme points belong to P. In a convex partition, the union of the internal faces give the convex hull of P and the interiors of the polygons are pairwise disjoint. Moreover, no polygon is allowed to contain a point of P in its interior. The problem is to find a convex partition with the minimum number of internal faces. The problem has been shown to be NP-hard and was recently used in the CG:SHOP Challenge 2020. We propose a new integer linear programming (IP) formulation that considerably improves over the existing one. It relies on the representation of faces as opposed to segments and points. A number of geometric properties are used to strengthen it. Data sets of 100 points are easily solved to optimality and the lower bounds provided by the model can be computed up to 300 points.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases convex partition, integer programming, geometric optimization

Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.20

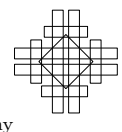
Related Version *Previous Version*: https://arxiv.org/abs/2012.07939

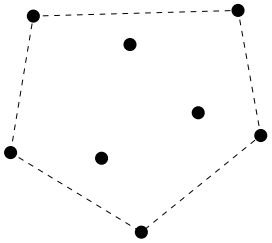
Funding *Nicolas Catusse*: Supported in part by the ANR Project DISTANCIA (ANR-17-CE40-0015) operated by the French National Research Agency (ANR).

1 Introduction

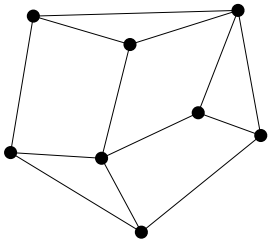
Let P be a set of points in the plane and $n = |P|$ the number of points. Let $H(P)$ denote the convex hull of P and $I(P)$ the set of internal points of P, i.e. the subset of points that are not vertices of the convex hull $H(P)$. A simple polygon is empty if it does not contain a point of P in its interior. A convex partition of P is a planar subdivision of $H(P)$ into non-overlapping empty convex polygons whose vertices are the points of P. The minimum convex partitions (MPC) problem is to find the convex partition minimizing the total number of empty convex polygons. In the remainder of the paper, we refer to such empty convex polygons as convex faces or simply **faces**. Fig. 1 illustrates the input data, its convex hull (1a) and two feasible solutions ((1b) and (1c)). Solution (1c) is using 5 faces which is optimal for this input data P. A practical application of this problem is reported in the area of network design [10]. Additionally, the authors of [2] argue that decomposition into polygons plays a key role in algorithm design where divide and conquer schemes take advantage of such decomposition to reduce the problem's size.

This problem was the subject of the 2020 Computational Geometry Challenge [4], which proposed a number of instances of size $n \in [10; 1000000]$ to be solved with no time limit [3]. A proof of NP-hardness for the case of planar point sets in not necessarily general position (three points can be collinear) has been announced by N. Grelier in [8]. The complexity of the MCP for points in general position (no three points are collinear) is still open. If the point sets can be decomposed into a constant number of convex layers, a polynomial-time algorithm

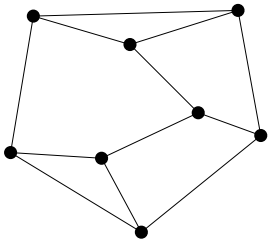


(a) Point set P with convex hull in dashed line.



(b) A feasible solution.



(c) The optimal solution.

Figure 1 Example of an instance and two solutions.

is given by Fevens, Meijer and Rappaport in [6]. For instance in general position, Knauer and Spillner gave a 3-approximation algorithm that runs in $O(n \log n)$, and a $\frac{30}{11}-$approximation of complexity $O(n^2)$ in [10].

The worst-case bound for a set of n points in general position has also been studied, see [9, 10, 11]. J. Urrutia conjectures that this number is at most $n+1$ in [13]. At the present time, the best known upper bound is $\frac{4}{3}n-2$ in [12]. Conversely, the best lower bound is $\frac{12}{11}n-2$ in [7].

For the exact resolution of the problem, in addition to the model of Barboza and al. [2] that we detail below, Da Wei Zheng et al. (winners of the 2020 CG challenge) proposed the use of a SAT model for the exact resolution of instances with $n \leq 50$ [14].

We propose a new formulation in integer linear programming[1]. It involves an exponential number of variables but $O(n^2)$ constraints. It considerably improves the formulation proposed by Barboza and al.[2] and allows to entirely close the corresponding benchmark [1]. The key idea of our approach is to reason with convex faces rather than edges and points.

Let us first introduce the notations as well as the formulation given in [2]. Let $\overline{ij}$ be the line segment between two distinct points $i, j \in P$ and S the set of all line segments of P. Let $E(P)$ be the set of edges corresponding to S, i.e. $E(P) = \{\{i,j\} \mid \overline{ij} \in S\}$. So the graph $G = (P, E(P))$ is a complete graph induced by P. The formulations presented below require to distinguish two *sides* to an edge $\{i,j\}$ namely the *left* and *right* side. For this, we set an orientation to the segments of $E(P)$ and define $A(P)$ the set of arcs (i,j) such that $\{i,j\} \in E(P)$ and the two points i and j are sorted by x-coordinate, i.e. $x_i < x_j$ then y-coordinate in case of a tie. Then we can define the orientation of another point $k \in P$ with respect to the arc (i,j) by the cross product: k is said to be to the left of (i,j) if $\vec{ij} \times \vec{ik} \geq 0$ (or $k \in \text{CCW}(ij)$), and to the right if $\vec{ij} \times \vec{ik} < 0$ ($k \in \text{CW}(ij)$). In the following, we will sometimes refer either to the edge $\{i,j\}$ or to the arc (i,j), notice that an arc always has a corresponding edge. A face f can be seen as a sub-graph $G_f = (V_f, E_f)$ of G and we refer to V_f and E_f respectively as the vertices and edges of the face f. Consider a convex face f and an edge $\{i,j\} \in E_f$ of its boundary. f is said to be to the right (resp. left) of the arc $(i,j) \in A(P)$ if for any vertex $k \in V_f$, k is at the right (resp. left) of (i,j). Since f is convex, all vertices of V_f share the same orientation with respect to (i,j). Finally, for each edge $\{i,j\}$, we define the set $R(i,j)$ (resp. $L(i,j)$) as the set of all convex faces to the right (resp. left) of arc (i,j). We denote by F the set of all possible empty convex faces for P, see Fig. 2 for an example.

[1] See the video for a didactive synthesis: `https://youtu.be/J9cW0vYaNzE`.

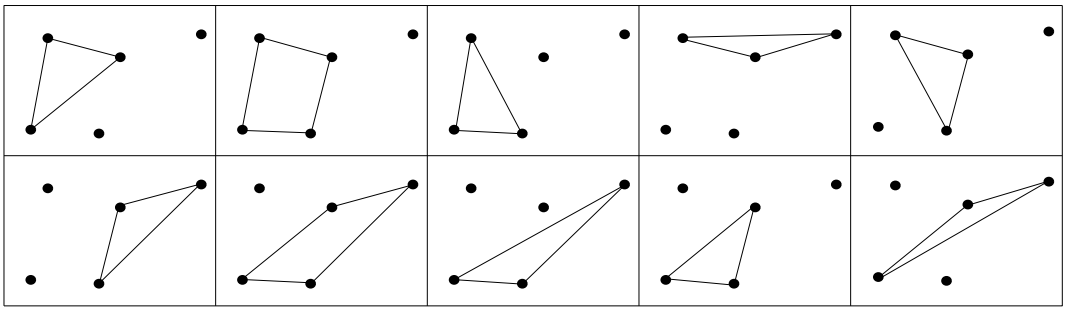


Figure 2 All possible faces F for an instance of P with $n = 5$.

Let us first recall the formulation proposed by Barboza and al. [2]. This formulation addresses the problem where the set P is in a general position, i.e. with no three points being collinear. It is a compact formulation with a boolean variable per edge which indicates if this edge is selected in the partition. Let $S^C \subseteq S$ be the set of pairs of crossing line segments and $E^C \subseteq E(P)$ the corresponding edges.

$$
\begin{aligned}
z^* = \text{minimize} \quad & \sum_{(ij)\in E(P)} x_{ij} && && (1)\\
\text{s.t.} \quad & x_{ij} + x_{kl} \leq 1 && \forall \{\{i,j\},\{k,l\}\} \in E^C && (2)\\
& x_{ij} = 1 && \forall \{i,j\} \in H(P) && (3)\\
& \sum_{k\in CCW(ij)} x_{ik} \geq 1 && \forall (i,j) \in A(P), i \in I(P) && (4)\\
& \sum_{j\in P} x_{ij} \geq 3 && \forall i \in P && (5)\\
& x_{ij} \in \{0,1\} && \forall \{i,j\} \in E(P) && (6)
\end{aligned}
$$

The objective function (1) is expressed in terms of number of edges, which is equivalent to the expression in terms of number of faces for this problem by Euler's formula. Planarity is ensured by the constraint (2). Constraint (3) imposes the selection of the edges of the convex hull $H(P)$ in the solution. Constraint (4) ensures local convexity for each internal point $i \in I(P)$, by forcing the selection of an edge to the left of each possible arc (i, j). The last constraint (5) is redundant with the previous convexity constraint (4), and is there to strengthen the formulation and improve the linear relaxation of the model.

A benchmark is introduced in [2] with two sets of instances, one with a number of points $n \leq 50$ (generated by keeping only instances for which the model above is able to prove optimality in 20 minutes in order to have a known set of optimal solutions), and the other with a larger number of points: $55 \leq n \leq 110$.

The paper is organized as follows. Section 2 presents a number of geometric results underlying the model proposed. Section 2 gives the novel integer programming formulation and Section 3 reports some experimental results.

2 Geometric observations related to convex faces

We review a number of observations that will help to establish and strengthen the formulation proposed Section 3.1. Note first that a very simple lower bound on the number of faces can be obtained using Euler's result. Since the edges of a convex partition form a planar graph, the well-known Euler formula holds and state that $\mathrm{f} = \mathrm{e} - \mathrm{v} + 1$ (without the external face) in any convex partition (f is the number of faces, e the number of edges and v is the number of

vertices). Moreover, in a convex partition, the convexity constraint requires that the angles between the incident edges of an internal point be less than or equal to 180 degrees. So the internal points can have a degree two in the collinear case, or at least three if they are not collinear with two other points. The points of $H(P)$ have a degree of at least two with their neighbors in the convex hull. Some vertices can be shown to have a greater degree (see Section 2.1 and Remark 3) and for each point $p \in P$, we can consider a lower bound $d(p)$ on its degree. Using Euler's formula, we can compute the following lower bound:

$$\mathrm{f} \geq \frac{1}{2}(\sum_{p \in P} d(p)) - n + 1$$

We refer to this lower bound as the Euler bound.

2.1 Geometric cuts

In general position, Knauer and al. in [10] proposed an interesting lower bound on the degrees of the vertices of $H(P)$. They first compute the convex hull of the internal points $H(I(P))$. According to the geometrical configuration of these points, they determine whether one or two edges are needed to connect them to the vertices of the convex hull $H(P)$. In addition, they present examples showing that this bound is almost tight.

We propose a generalization of this bound. Let $\mathscr{C}$ be a convex subset of $\mathbb{R}^2$, $P^{\mathscr{C}}$ the points of P inside $\mathscr{C}$ and $\overline{P^{\mathscr{C}}} = P \setminus P^{\mathscr{C}}$ its complement. We will determine $d(P^{\mathscr{C}})$, a lower bound of the number of edges between $P^{\mathscr{C}}$ and $\overline{P^{\mathscr{C}}}$.

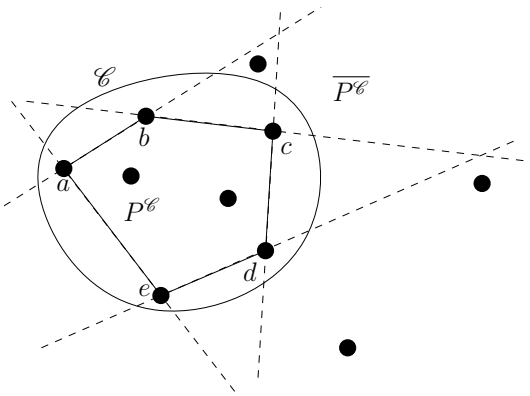
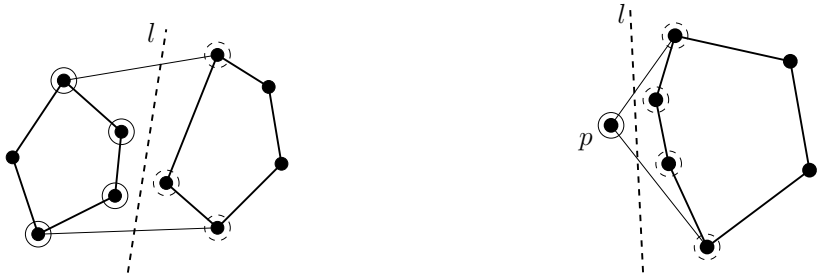


Figure 3 An example of partition with a convex $\mathscr{C}$.

Let $P'^{\mathscr{C}}$ be the subset of points $p \in H(P^{\mathscr{C}})$ such that there exists a line segment $\overline{pp'}$ with a point $p' \in \overline{P^{\mathscr{C}}}$ that does not cross $H(P^{\mathscr{C}})$, i.e. such that $\overline{pp'} \cap H(P^{\mathscr{C}}) = p$. In terms of the visibility graph, if we consider that $H(P^{\mathscr{C}})$ is an obstacle, for each point $p \in P'^{\mathscr{C}}$, there exists an edge in the visibility graph between p and a point $p' \in \overline{P^{\mathscr{C}}}$.

To deal with collinearity, we now remove from $H(P^{\mathscr{C}})$ the vertices of the convex hull which are collinear with their two neighbors. As in [10], we classify the vertices of $P'^{\mathscr{C}}$ into distinct sets. Let v be a point of $P'^{\mathscr{C}}$ and u and w its neighbors in the convex hull $H(P^{\mathscr{C}})$. Let $\mathcal{H}_w$ (resp. $\mathcal{H}_u$) be the half-plane bounded by the straight line passing through v and u (resp. w) and not containing w (resp. u). The vertex v is of type (1) if $\mathcal{H}_w \cap \mathcal{H}_u$ contains at least one point of $\overline{P^{\mathscr{C}}}$, type (2) if $v \notin H(P)$ and $\mathcal{H}_w \cap \mathcal{H}_u$ contains no point of $\overline{P^{\mathscr{C}}}$, and type (3) if $v \in H(P)$. Let n_1, n_2 and n_3 be respectively the number of points of type (1), (2) and (3). Fig. 3 shows an example of partition with a convex $\mathscr{C}$. The points $\{a, b, c, d, e\}$



(a) (b)

Figure 4 Two partition of P by a line l, P_1^l is circled solid and P_2^l is circled dashed.

belong to the convex hull $H(P^{\mathscr{C}})$. Point $a \notin P'^{\mathscr{C}}$ because there is no line segment between a and a point of $\overline{P^{\mathscr{C}}}$. So $P'^{\mathscr{C}}$ is the set $\{b, c, d, e\}$. The point d is of type (1), c is of type (2), and b, e are of type (3) because they belong to $H(P)$.

▶ **Lemma 1.** *In any solution, the number of edges between $P^{\mathscr{C}}$ and $\overline{P^{\mathscr{C}}}$ is at least $n_1+2n_2+n_3$, $d(P^{\mathscr{C}}) \geq n_1 + 2n_2 + n_3$.*

Proof. A vertex $v \in P'^{\mathscr{C}}$ cannot satisfy its local convexity constraint only with the vertices of $H(P^{\mathscr{C}})$ since no three points of $H(P^{\mathscr{C}})$ are on the same line and there is a half-plane passing through v containing all the vertices of $H(P^{\mathscr{C}})$. So there must be at least an edge between v and a point of $\overline{P^{\mathscr{C}}}$. For type (1), there is a point $p' \in \overline{P^{\mathscr{C}}}$ in $H_w \cap H_u$ such that the edge $\{v, p'\}$ can satisfy the convexity constraint. For type (2), at least two edges are needed. Vertices of $H(P)$ (type (3)) don't have to satisfy the convexity constraint, and need only the edge of $H(P)$ between $P^{\mathscr{C}}$ and $\overline{P^{\mathscr{C}}}$, which are mandatory. ◀

A simple way to divide P into two convex partitions is to use a straight line to partition the points of P. We propose a lower bound on the minimum number of edges that cross a straight line in any solution.

▶ **Corollary 2.** *Let l be a straight line that divides P into two parts P_1 and P_2, and let $H(P_1)$ and $H(P_2)$ be their respective convex hulls (see Fig. 4). Let P_1^l (resp. P_2^l) be the set of point $p \in H(P_1)$ (resp. $H(P_2)$) such that there exists a line segment with a point $p' \in P_2$ (resp. P_1) and $\overline{pp'}$ does not cross $H(P_1)$ (resp. $H(P_2)$), i.e. $\overline{pp'} \cap H(P_1) = p$ (resp. $\overline{pp'} \cap H(P_2) = p$). In any solution, there is at least $\max(d(P_1^l), d(P_2^l))$ edges that cross the line l.*

Proof. We simply apply the Lemma 1 with P_1 and with P_2, and take the maximum of the two lower bounds. ◀

▶ Remark 3. With Corollary 2, we obtain a minimum bound on the degree of the points belonging to the convex hull $H(P)$. Indeed, by definition of the convex hull, for a point $p \in H(P)$ we can always partition P with a straight line into two sets $\{p\}$ and $\{P \setminus p\}$, so $d(p)$ is obtained by applying the Lemma 1 on $P \setminus \{p\}$.

Fig. 4b shows an example of this lower bound on the degree of a point in $H(P)$. The leftmost point p has at least 4 edges in any solution, so $d(p) = 4$.

2.2 Mandatory face

Recall that F is the set of all faces of P. Among this set, if a face must be present in any feasible solution, it is considered as mandatory.

▶ Remark 4. If a point of $\mathbb{R}^2$ inside the convex hull $H(P)$ is covered by only one convex $f \in F$, f must be in the solution.

Indeed, since any point of $\mathbb{R}^2$ inside the convex hull $H(P)$ must belong to a face of the partition solution, the only face f that contains this point must be part of the solution.

An example of a mandatory face is shown in Fig. 5. The edge $\{i, j\}$ belongs to the convex hull, so it is mandatory. Let us now observe that taking the line passing through $\{j, k\}$, the side containing the $\{i, j\}$ edge contains no other point than i. Similarly, taking the line passing through $\{i, k\}$, the side containing the edge $\{i, j\}$ does not contain any other point than j. Therefore, no other convex face than the face defined by vertices $\{i, j, k\}$ contains the segment $\{i, j\}$.

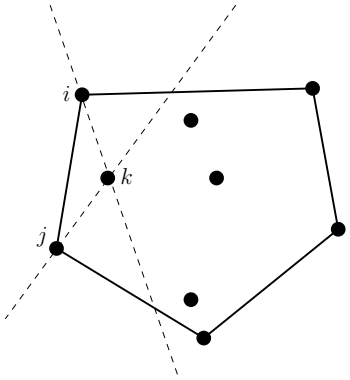


Figure 5 Example of a mandatory face $\{i, j, k\}$.

2.3 Dominated face

Conversely, among the set F, one can determine a set of polygons which are sufficient to obtain an optimal solution. Such a set is called a dominant set. We are now going to describe several sets of faces that can be eliminated from the faces to be considered without losing any optimal solution.

Two faces f_1 and f_2 are said to be adjacent along an edge e if $E_{f_1} \cap E_{f_2} = e$. The union $f_1 \cup f_2$ of two adjacent faces f_1 and f_2 is a polygon defined by the union of their surfaces and the removal of the common edge e, i.e. $E_{f_3} = E_{f_1} \cup E_{f_2} \setminus \{e\}$. The resulting polygon f_3 is not necessarily convex.

▶ Remark 5. A face f is dominated if for all f' adjacent to f, $f \cup f'$ is convex (is in F).

▶ Remark 6. If an edge belongs only to dominated faces, this edge does not belong to an optimal solution. Such an edge is called a dominated edge.

For instance, an edge of two non-consecutive vertices of the convex hull is a dominated edge. The points of the convex hull locally satisfy their convexity constraint with the edges of the convex hull which are mandatory. So the union between two faces that share an edge between two vertices of the convex hull is always convex and a solution without this edge is always better (see Fig. 6).

2.4 Generation of the convex faces set

In Section 3.1, our integer linear program requires the computation of F, i.e. the enumeration of the possible faces for P. See Fig. 2 for an example of set F ($n = 5$). We rely on the article of Dobkin and al. [5] which describes an algorithm to find the set of empty convex r-gons (r

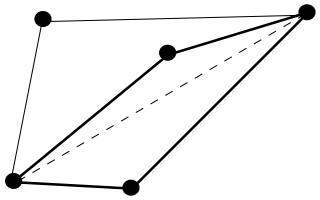


Figure 6 From example of Fig. 2, the dashed edge is dominated, as well as the two triangle faces that contain it.

is the number of points of the polygon) for a set of points in general position. The principle of the algorithm is as follows. For each point $p \in P$, a star-shaped polygon is formed by sorting the vertices to the left of p by angle around p. We then compute the visibility graph of this polygon, including the edges of the polygon, but removing the edges containing p. We finally compute the set of convex chains in the visibility graph (in [5] the algorithm lists the convex chains of $r-2$ edges, but here we want all the faces of any size). For each chain, by adding the two edges connecting the ends with p, we obtain a face. We have adapted this algorithm to handle the case of collinear points.

With the properties seen in this section, we can preprocess the F set to eliminate some of the dominated faces, we call this subset $F^{PP} \subseteq F$ (PP for PreProcessing). In practice, filtering all the dominated faces can be very expensive. We chose to perform an incomplete but efficient preprocessing with the following algorithm.

Algorithm 1 Faces pre-processing.

1: $F^{PP} \leftarrow F$
2: **for** each edge $\{i,j\}$ **do**
3: **if** $\min(|R(i,j)|, |L(i,j)|) \leq 3$ **then**
4: **if** the union of any two faces of $R(i,j) \times L(i,j)$ is convex **then**
5: Remove all faces using edge $\{i,j\}$ from F^{PP}.
6: **for** each face $f \in F^{PP}$ and each edge $\{i,j\}$ of E_f **do**
7: **if** the union of f with all its adjacent faces along $\{i,j\}$ on the other side (if $f \in R(i,j)$, all faces of $L(i,j)$, otherwise all faces of $R(i,j)$) is convex **then**
8: Remove the face f from F^{PP}.
9: **for** each edge $\{i,j\}$ s.t i and j are non-consecutive vertices of the convex hull $H(P)$ **do**
10: Remove all faces using edge $\{i,j\}$ from F^{PP}.

Note that the condition at line 3 deliberately limits the convexity verification of the union of $R(i,j) \times L(i,j)$ faces to cases where $|R(i,j)|$ or $|R(i,j)|$ is less than three for efficiency reasons. Note also that we are only deleting faces. Indeed, the starting set F is made by definition of all the possible faces so when we check if the union of two faces is convex, the resulting face of this union is already in F. Finally, this algorithm remains exponential in the worst case due to line 7 but is efficient enough in practice.

3 An IP formulation based on convex polygons

3.1 Formulation

Recall that F denotes the set of all convex faces. Consider a variable $x_f \in \{0,1\}$ for each face $f \in F$ that is set to 1 if f is in the convex partition, and 0 otherwise.

We propose the following Integer Programming model:

$$\text{minimize } z = \sum_{f \in F} x_f \tag{1}$$

$$(M) \qquad \sum_{f \in L(i,j)} x_f - \sum_{f \in R(i,j)} x_f = \begin{cases} 1 & \forall \{i,j\} \in H(P),\ L(i,j) \neq \emptyset \\ -1 & \forall \{i,j\} \in H(P),\ R(i,j) \neq \emptyset \\ 0 & \forall \{i,j\} \in E \setminus H(P) \end{cases} \tag{2}$$

$$x_f \in \{0,1\} \qquad \forall f \in F \tag{3}$$

The objective function (1) minimizes the total number of faces chosen for the solution. Constraints (2) ensure that for each edge of $H(P)$, there is a face chosen at the left or right, depending on the orientation of the edge, in the solution. It also ensures that for all remaining internal edges, the number of faces chosen on its right is equal to the number of faces chosen on its left (this number will be either 0 or 1). The constraint is very similar to a flow or path conservation in a network and can be understood as a *conservation of faces*, i.e. that whenever a face is used on one side of an edge, another face must match the other side (to the exception of hull which acts as a *source* or *sink* of faces). It enforces the faces chosen to tile the interior of $H(P)$. Note that the model is however not a network flow model.

Although this formulation is correct as it is, it can be strengthened by enforcing the minimum number of convex faces, denoted $d(i)$ that are adjacent to each point $i \in P$ (the degree of a vertex). Note that $d(i) \geq 2$ using the reasonings presented in Section 2.1 and in particular Remark 3.

$$\sum_{f | i \in V_f} x_f \geq d(i) \quad \forall i \in P \tag{4}$$

Finally note that the linear relaxation consists in relaxing constraint (3) into $x_f \geq 0$ and the optimal value is denoted z^*_{LP}.

Cutting planes. Observe that a feasible integer solution of (M) is an independent set in the intersection graph of the convex faces. More precisely, let $I = (V_F, E_F)$ be an undirected graph where each vertex $v_f \in V_F$ is associated to a convex face $f \in F$ and an edge (v_a, v_b) is added to E_F if faces a and b intersect. Any feasible solution to (M) is an independent set of I although the converse is not true. As a result, a number of valid inequalities known for independent set can be used to strengthen (M). Since the size of I is exponential in n, such inequalities must be added as cutting planes by solving a separation problem in the solution of the linear relaxation. We considered two classes of such inequalities, the clique and the odd cycle inequalities. Let $C \subseteq V_F$ be a clique of I, i.e. a set of two by two intersecting faces. Since only one face of C can be chosen in a solution, the following cutting plane can be added.

$$\sum_{f \in C} x_f \leq 1$$

Let $O \subseteq V_F$ be an odd cycle of I, the following inequality holds:

$$\sum_{f \in C} x_f \leq \frac{1}{2}(|O| - 1).$$

Size of the formulation and preprocessing. The enumeration of all convex faces required by (M) is discussed Section 2.4. It is easy to see that the number of convex faces grows exponentially with the number of points. The worst case is hit when all points belongs to the

convex hull. In this case, any subset of points defines a valid convex face leading to 2^n convex faces. In general, $|F| \in O(2^n)$ and formulation (M) has an exponential number of variables. In practice, some points lie inside the hull and not all subsets of points define a convex face. We will see Section 4 that the formulation scales well to practical instances of size $n = 100$. After preprocessing of the faces, the formulation is stated on the set F^{PP} and the mandatory faces f (See Remark 4) are enforced with $x_f = 1$. The number of constraints is in $O(n^2)$ and the preprocessing also helps reducing this number in practice (See Remark 6).

4 Experimental results

We report the experimental evaluation of the proposed model. Let us first give details on the hardware, software as well as the benchmark used.

Computational Environment. All experiments were run on a Macbook pro with 4 processor cores Intel Core i7 at 2.8 GHz and a limit of 4 Go of RAM. Algorithms are all implemented in Java and CPLEX 12.8.0 is used in multi-thread mode. Note that experiments of Barboza and al [2], which serve as a baseline, were run in single thread mode.

Benchmark. Two sets of instances $T1$ and $T2$ are available from [1]. $T1$ is made of 30 instances for each size n in $30, 32, \ldots, 50$ that is a total of 330 instances. Each instance is generated using a uniform random distribution of the coordinates in the interval $[0, 1]$. But for each size, the first 30 instances that were found optimally solvable within 20 minutes were kept. Some instances were rejected (hit the time limit) from size $n = 44$ and onwards so that set $T1$ is biased to be easy enough for the model given in [2]. Set $T2$ is made of 30 instances for each sizes $55, 60, \ldots, 110$ (so 330 in total) and no selection was made on set $T2$. A set of more *structured* instances was proposed in the CG:SHOP challenge 2020 [3, 4]. We use instances of sizes ≤ 100 from the image set. This leads to four instances (four images: euro-night, london, stars, us-night) for each size n in $10, 15, 20, \ldots, 100$ so a total of 56 instances. We also report some results on the 20 instances of size $100 < n \leq 300$ of the challenge[2].

The results are presented in two parts. Firstly, we evaluate our model by solving the integer model. Secondly, we discuss the lower bound that can be obtained for larger problem of sizes $150 < n \leq 300$ by focusing on the linear relaxation.

4.1 Exact solving

Table 1 and 2 presents the results on two benchmarks. Each line gives a summary of the results for a class of instances gathered by size n and column # gives the number of instances considered in the class. Note that all instances of size $n \leq 50$ are considered in a single class (the first line). For each class of instances, we report a number of metrics. We report the average number of faces ($Avg\ |F|$) as well as the number of faces after preprocessing using Algorithm 1 ($Avg\ |F^{PP}|$). We solve the linear relaxation z^*_{LP} of our model [3] (changing all $x_f \in \{0, 1\}$ into $x_f \geq 0$) and reports the median (Med), average (Avg) and maximum (Max) gap (column $\%Gap$) computed as follows: $100(z^* - \lceil z^*_{LP} \rceil)/z^*$. We also report the median, average and maximum solving time ($Cpu(s)$) in seconds. Finally, the maximum number of nodes ($\#\ Nodes$) explored by the branch and bound algorithm is given.

[2] All the instances used as well as the detailed results are available on the webpage `https://pagesperso.g-scop.grenoble-inp.fr/~cambazah/convexpartition/minconvex.html`

[3] Note that this is the linear relaxation and not the root node lower bound that is usually much stronger after the automatic preprocessing performed by cplex.

Table 1 Experimental results on the entire benchmark T1 and T2 of Barboza and al [2]. All instances are solved to optimality.

#	n	Faces		%Gap			Cpu(s)			#nodes
		Avg $\|F\|$	Avg $\|F^{PP}\|$	Med	Avg	Max	Med	Avg	Max	Max
330	30-50	11417	5592	0,00	0,02	3,85	0,35	0,41	1,58	0
30	55	23167	12253	0,00	0,20	3,13	0,87	1,07	1,86	0
30	60	27974	15141	0,00	0,00	0,00	1,11	1,35	2,48	0
30	65	35061	19201	0,00	0,17	2,56	1,79	1,89	3,00	0
30	70	40933	22392	0,00	0,32	2,56	2,46	2,31	6,13	45
30	75	48998	27818	0,00	0,44	2,38	3,55	3,91	9,37	25
30	80	55737	31606	0,00	0,41	2,08	4,09	3,71	6,61	0
30	85	66366	37677	0,00	0,48	2,17	5,31	5,45	13,18	26
30	90	73386	42299	0,00	0,51	1,92	5,74	7,49	35,18	77
30	95	84009	48642	0,00	0,90	3,51	8,83	16,23	82,61	1913
30	100	95135	55078	0,00	0,47	1,85	8,60	13,58	62,84	125
30	110	116568	69006	1,55	1,00	3,08	18,16	31,78	166,73	2399

- All instances of size $n \leq 50$ are easily solved optimally with a maximum time of 1,58 seconds (Table 1). Note the none of the instances require branching from the solver, i.e. that all are solved at the root node.
- The approach remains very efficient even for instances of size up to $n = 110$ with an average time 31,78 seconds (Table 1). Branching is sometimes required from size $n = 75$ and onwards but the quality of the linear relaxation appears to be very good with a maximum gap of 3.51% across all instances of size $55 \leq n \leq 110$. The median gap is null (except for $n = 110$) showing that the linear relaxation gives the optimal value for most of the instances.
- The preprocessing of faces remove around 44% of them in average across all instances. However it seems to decrease slowly as the size increases and is around 41% for instances of size $n = 100$ (Table 1).
- The results observed on the structured image instances (Table 2) of the challenge are very similar and all instances are easily solved to optimality.
- Table 3 gives the results for instances with numerous collinear points on vertical and horizontal lines. Although these structured instances tend to have a large number of faces, the model can solve optimally the three instances available with a size around 100. Considering the large number of faces involved, we also give the results without pre-processing of the faces to show its interest.

Even though the solving times can not be directly compared to [2] due to different hardware and number of threads, we believe it is a significant improvement. Some of the instances of size ≤ 50 could require a computation time of 20 minutes and none of instances with $55 \leq n \leq 100$ can be solved exactly in [2] where elaborate heuristics are used instead.

We now turn our attention to larger instances and discuss the possibility to use model (M) to produce a lower bound.

Table 2 Experimental results on the images benchmark ($n \leq 100$) of the challenge. All instances are solved to optimality. Each class is made of four instances referred to as euro-night, london, stars and us-night.

#	n	Faces		%Gap			Cpu(s)			#nodes
		Avg $\|F\|$	Avg $\|F^{PP}\|$	Med.	Avg	Max	Med	Avg	Max	Max
4	10	141	46	0,00	0,00	0,00	0,00	0,00	0,01	0
4	15	532	186	0,00	0,00	0,00	0,01	0,01	0,03	0
4	20	1308	659	0,00	0,00	0,00	0,03	0,04	0,07	0
4	25	2717	1276	0,00	0,00	0,00	0,06	0,06	0,06	0
4	30	3921	2170	0,00	0,00	0,00	0,11	0,11	0,15	0
4	35	6455	4047	0,00	0,00	0,00	0,15	0,14	0,16	0
4	40	11080	5627	0,00	0,00	0,00	0,36	0,38	0,61	0
4	45	12989	6702	1,79	1,82	3,70	0,42	0,50	0,86	0
4	50	20686	11826	0,00	0,00	0,00	1,18	1,42	2,44	0
4	60	26780	16091	0,00	0,00	0,00	1,77	1,63	2,25	0
4	70	39468	24258	0,00	0,00	0,00	1,35	1,58	2,44	0
4	80	53071	34618	1,00	1,04	2,17	5,41	5,83	8,86	0
4	90	76710	47341	0,00	0,00	0,00	7,26	6,40	7,79	0
4	100	91813	53815	0,00	0,43	1,72	11,53	17,55	42,53	63

Table 3 Experimental results on the instances ($100 < n < 150$) of the challenge with preprocessing (first three lines) and without preprocessing (three last lines). The three instances are solved to optimality.

name	n	$\|F\|$	$\|F^{PP}\|$	Pp(s)	Eul	$\lceil z^*_{LP} \rceil$	z^*	%Gap	Cpu(s)	#nodes
rop101	101	449341	326348	16,43	4	21	21	0	177,14	10
rop107	107	785030	620263	262,03	4	23	24	4,17	1195,59	80
rop122	122	703772	553370	30,68	3	22	23	4,35	748,73	109
rop101	101	449341	-	0,39	4	21	21	0	222,53	14
rop107	107	785030	-	0,6	4	23	24	4,17	1389,5	90
rop122	122	703772	-	0,81	3	22	23	4,35	1495,05	67

4.2 Lower bounds

We now consider the linear relaxation z^*_{LP} of model (M) on larger instances. Table 4 reports the results obtained on all instances of the challenge [3] of sizes $150 < n \leq 300$[4]. For each instance, the table gives the best known upper bound obtained during the challenge (UB), the number of convex faces remaining after preprocessing ($|F^{PP}|$), the percentage of reduction compared to the initial number of faces ($\%RF$) computed as $100(1 - \frac{|F^{PP}|}{|F|})$, the percentage of reduction of the number of edges ($\%RE$) computed as $100(1 - \frac{|E^{PP}|}{n(n-1)/2})$ the cpu time in seconds required for pre-processing ($Pp(s)$), the value of the Euler lower bound (Eul), the

[4] us-night-0000200, paris0000200, stars0000200, uniform00002001, uniform00002002, euronight0000200, ortho_rect_union_170, ortho_rect_union_186, ortho_rect_union_199, ortho_rect_union_208, rop0000262, us-night0000300, paris0000300, stars0000300, uniform0000300-1, uniform0000300-2, euro-night0000300

Table 4 Lower bounds ($\lceil z^*_{LP} \rceil$) computed on all instances of the challenge with ($150 < n \leq 300$).

			Face generation					Linear relaxation			
name	n	UB	$\lvert F^{PP} \rvert$	%RF	%RE	Pp(s)	Eul	$\lceil z^*_{LP} \rceil$	%Gap	Slv(s)	Tot(s)
us	200	107	298473	37,2	36,1	4,5	97	105	1,87	19,7	25,4
paris	200	110	259023	36,6	36,4	5,3	99	108	1,82	15,9	22,2
stars	200	110	285256	37,5	35,5	1,6	100	109	0,91	23,4	26,0
unif-1	200	112	271086	36,9	36,4	2,3	100	110	1,79	16,5	19,8
unif-2	200	111	280183	37,7	36,9	3,0	98	109	1,80	16,1	21,8
euro	200	110	263472	35,5	35,8	2,2	98	108	1,82	16,5	19,6
ortho	170	62	404356	30,1	35,6	5,4	10	59	4,84	10,8	17,7
ortho	186	65	469839	28,3	33,9	4,1	11	60	7,69	20,3	26,0
ortho	199	71	527131	30,7	34,6	11,6	11	66	7,04	22,7	36,3
ortho	208	74	703891	24,4	34,4	8,3	10	69	6,76	25,2	36,3
rop	262	41	3225861	15,9	22,2	211,4	3	40	2,44	342,3	574,6
us	300	160	760641	34,2	35,4	4,9	149	158	1,25	88,1	95,7
paris	300	161	651331	34,8	36,3	4,7	149	157	2,48	82,9	90,0
stars	300	163	760654	33,8	35,4	6,6	149	160	1,84	101,3	110,7
unif-1	300	161	654205	36,9	36,6	7,1	143	158	1,86	81,8	91,8
unif-2	300	167	640052	36,2	37,0	4,6	155	163	2,40	74,4	81,5
euro	300	163	776699	33,6	35,7	6,4	147	158	3,07	84,8	94,0

value of the linear relaxation of the proposed model ($\lceil z^*_{LP} \rceil$), the gap to the best known upper bound (*%Gap*) computed as $100(UB - \lceil z^*_{LP} \rceil)/UB$, the time for computing the relaxation (*Slv(s)*) in seconds and the total time in seconds (*Tot(s)*).

- Despite the large number of convex faces, the model scales much better than expected and solving the linear relaxation itself is faster than the preprocessing of faces.
- The number of convex faces is considerably larger for the instances referred to as *ortho* or *rop* where the points of P are collinear on vertical and horizontal lines. Note that Euler's formula lead to a very weak bound in this latter case. Note also that more than a third of the edges are often removed by preprocessing.
- The lower bound computed improves significantly the bound provided by the Euler's formula with gaps less than 3% for the image benchmark and gaps less than 8% for the orthogonal benchmark.

Experiments using the cutting planes mentioned remained unconclusive so far and are not reported in details in the present paper. The cutting planes seem to provide small improvements of the bound and the separation routine is computationally expensive.

5 Conclusion and future work

We propose a new formulation in integer linear programming for the minimum convex partition problem. It proves able to solve to optimality all instances of less than a hundred points proposed so far in the literature, considerably improving the formulation given by [2]. Despite the exponential number of variables involved, its linear relaxation can be solved efficiently for instances of size up to 300 points providing strong lower bounds.

A first direction of research is to investigate further the cutting planes that could be used from the independent set formulation of the problem since a large family of such inequalities are already known. More interestingly, cutting planes can also be derived from geometric

statements. For instance, the result proposed in Lemma 1 directly lead to a cutting plane but require to study a separation algorithm. A second direction of research is to propose an implicit enumeration of the convex faces focusing on the faces with negative reduced cost only as opposed to generate and preprocess the entire set of faces. In other words, the linear relaxation of (M) could be solved using a column generation procedure.

References

1. Allan S. Barboza, Cid C. de Souza, and Pedro J. de Rezende. Minimum convex partition of point sets - benchmark instances and solutions, 2018. URL: `www.ic.unicamp.br/~cid/Problem-instances/Convex-Partition`.
2. Allan S. Barboza, Cid C. de Souza, and Pedro J. de Rezende. Minimum convex partition of point sets. In Pinar Heggernes, editor, *Algorithms and Complexity - 11th International Conference, CIAC 2019, Rome, Italy, May 27-29, 2019, Proceedings*, volume 11485 of *Lecture Notes in Computer Science*, pages 25–37. Springer, 2019.
3. Erik D. Demaine, Sándor P. Fekete, Phillip Keldenich, Dominik Krupke, and Joseph S. B. Mitchell. Cg challenge 2020 - minimum convex partition, 2020. URL: `https://cgshop.ibr.cs.tu-bs.de/competition/cg-shop-2020`.
4. Erik D. Demaine, Sándor P. Fekete, Phillip Keldenich, Dominik Krupke, and Joseph S. B. Mitchell. Computing convex partitions for point sets in the plane: The cg:shop challenge 2020, 2020. `arXiv:2004.04207`.
5. David P. Dobkin, Herbert Edelsbrunner, and Mark H. Overmars. Searching for empty convex polygons. *Algorithmica*, 5(4):561–571, 1990.
6. T. Fevens, H. Meijer, and D. Rappaport. Minimum convex partition of a constrained point set. *Discret. Appl. Math.*, 109:95–107, 2001.
7. J. García-López and C.M. Nicolás. Planar point sets with large minimum convex decompositions. *Graphs and Combinatorics*, 29:1347–1353, 2013.
8. Nicolas Grelier. Hardness and approximation of minimum convex partition, 2020. `arXiv:1911.07697`.
9. Kiyoshi Hosono. On convex decompositions of a planar point set. *Discrete Mathematics*, 309(6):1714–1717, 2009.
10. C. Knauer and A. Spillner. Approximation algorithms for the minimum convex partition problem. In *Scandinavian Workshop on Algorithm Theory (SWAT)*, 2006.
11. Mario Lomeli-Haro. Minimal convex decompositions, 2012. `arXiv:1207.3468`.
12. Toshinori Sakai and Jorge Urrutia. Convex decompositions of point sets in the plane, 2019. `arXiv:1909.06105`.
13. Jorge Urrutia. Open problem session. In *Canadian Conference on Computational Geometry (CCCG)*, 1998.
14. Da Wei Zheng, Jack Spalding-Jamieson, and Brandon Zhang. Computing Low-Cost Convex Partitions for Planar Point Sets with Randomized Local Search and Constraint Programming (CG Challenge). In Sergio Cabello and Danny Z. Chen, editors, *36th International Symposium on Computational Geometry (SoCG 2020)*, volume 164 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 83:1–83:7, Dagstuhl, Germany, 2020. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

Geometric Algorithms for Sampling the Flux Space of Metabolic Networks

Apostolos Chalkis
Department of Informatics and Telecommunications,
National and Kapodistrian University of Athens, Greece
Athena Research Innovation Center, Athens, Greece

Vissarion Fisikopoulos
Department of Informatics and Telecommunications,
National and Kapodistrian University of Athens, Greece

Elias Tsigaridas
Inria Paris and IMJ-PRG, Sorbonne Université, France
Paris Université, France

Haris Zafeiropoulos
Department of Biology, University of Crete, Heraklion, Greece
Institute of Marine Biology, Biotechnology and Aquaculture,
Hellenic Centre for Marine Research, Anavyssos Attiki, Greece

Abstract

Systems Biology is a fundamental field and paradigm that introduces a new era in Biology. The crux of its functionality and usefulness relies on metabolic networks that model the reactions occurring inside an organism and provide the means to understand the underlying mechanisms that govern biological systems. Even more, metabolic networks have a broader impact that ranges from resolution of ecosystems to personalized medicine.

The analysis of metabolic networks is a computational geometry oriented field as one of the main operations they depend on is sampling uniformly points from polytopes; the latter provides a representation of the steady states of the metabolic networks. However, the polytopes that result from biological data are of very high dimension (to the order of thousands) and in most, if not all, the cases are considerably skinny. Therefore, to perform uniform random sampling efficiently in this setting, we need a novel algorithmic and computational framework specially tailored for the properties of metabolic networks.

We present a complete software framework to handle sampling in metabolic networks. Its backbone is a Multiphase Monte Carlo Sampling (MMCS) algorithm that unifies rounding and sampling in one pass, obtaining both upon termination. It exploits an improved variant of the Billiard Walk that enjoys faster arithmetic complexity per step. We demonstrate the efficiency of our approach by performing extensive experiments on various metabolic networks. Notably, sampling on the most complicated human metabolic network accessible today, Recon3D, corresponding to a polytope of dimension 5 335, took less than 30 hours. To our knowledge, that is out of reach for existing software.

2012 ACM Subject Classification Mathematics of computing → Mathematical software; Applied computing → Systems biology; Computing methodologies → Modeling and simulation

Keywords and phrases Flux analysis, metabolic networks, convex polytopes, random walks, sampling

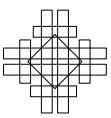
Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.21

Related Version *Full Version*: `https://arxiv.org/abs/2012.05503`

Supplementary Material *Software (Source Code)*:
`https://github.com/GeomScale/volume_approximation/tree/socg21`
archived at `swh:1:rev:96344048632acd7871b0a92ac903fc07575517f4`

Funding *Apostolos Chalkis*: Supported by PeGASUS.
Elias Tsigaridas: Partially supported by ANR JCJC GALOP (ANR-17-CE40-0009), the PGMO grant ALMA and the PHC GRAPE.

37th International Symposium on Computational Geometry (SoCG 2021).
Editors: Kevin Buchin and Éric Colin de Verdière; Article No. 21; pp. 21:1–21:16



Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Acknowledgements We would like to thank the anonymous reviewers for their helpful comments and suggestions. We also thank Ioannis Emiris for his useful comments.

1 Introduction

1.1 The field of Systems Biology

Systems Biology establishes a scientific approach and a paradigm. As a research approach, it is the qualitative and quantitative study of the systemic properties of a biological entity along with their ever evolving interactions [32, 33]. By combining experimental studies with mathematical modeling it analyzes the function and the behavior of biological systems. In this setting, we model the interactions between the components of a system to shed light on the system's *raison d'être* and to decipher its underlying mechanisms in terms of evolution, development, and physiology [27].

Initially, Systems Biology emerged as a need. New technologies in Biology accumulate vast amounts of information/data from different levels of the biological organization, i.e., genome, transcriptome, proteome, metabolome [49]. This leads to the emerging question *"what shall we do with all these pieces of information"?* The answer, if we consider Systems Biology as a paradigm, is to move away from reductionism, still the main conceptual approach in biological research, and adopt holistic approaches for interpreting how a system's properties emerge [43]. The following diagram provides a first, rough, mathematical formalization of this approach.

$$components \rightarrow networks \rightarrow in\ silico\ models \rightarrow phenotype\ [47].$$

Systems Biology expands in all the different levels of living entities, from the molecular, to the organismal and ecological level. The notion that penetrates all levels horizontally is *metabolism*; the process that modifies molecules and maintains the living state of a cell or an organism through a set of chemical reactions [53]. The reactions begin with a particular molecule which they convert into some other molecule(s), while they are catalyzed by enzymes in a key-lock relationship. We call the quantitative relationships between the components of a reaction *stoichiometry*. Linked reactions, where the product of the first acts as the substrate for the next, build up metabolic pathways. Each pathway is responsible for a certain function. We can link together the aggregation of all the pathways that take place in an organism (and their corresponding reactions) and represent them mathematically using the reactions' stoichiometry. Therefore, at the species level, metabolism is a network of its metabolic pathways and we call these representations *metabolic networks*.

1.2 From metabolism to computational geometry

The complete reconstruction of the metabolic network of an organism is a challenging, time consuming, and computationally intensive task; especially for species of high level of complexity such as *Homo sapiens*. Even though sequencing the complete genome of a species is becoming a trivial task providing us with quality insight, manual curation is still mandatory and large groups of researchers need to spend a great amount of time to build such models [57]. However, over the last few years, automatic reconstruction approaches for building genome-scale metabolic models [40] of relatively high quality have been developed. Either way, we can now obtain the metabolic network of a bacterial species (single cell species) of a tissue and even the complete metabolic network of a mammal. Biologists are also moving towards obtaining such networks for all the species present in a microbial

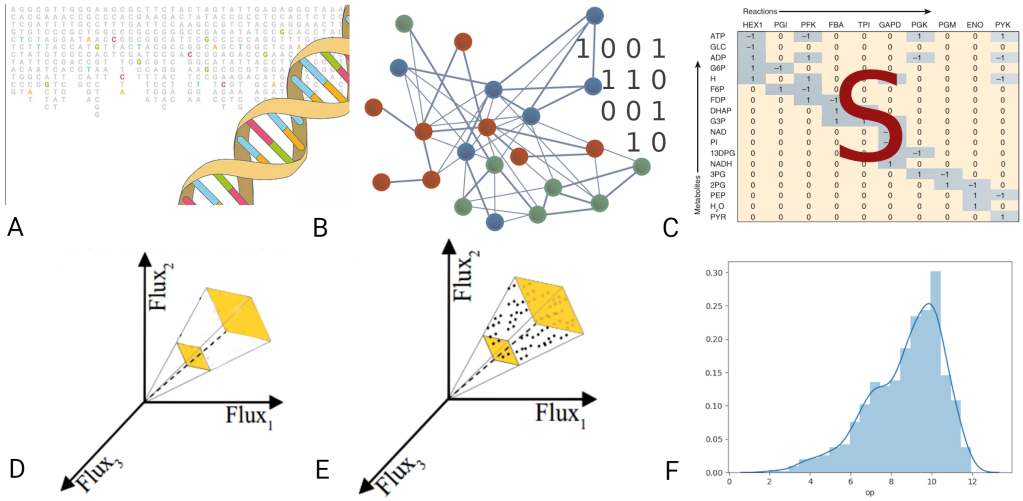


Figure 1 From DNA sequences to distributions of metabolic fluxes. (A) The genes of an organism provide us with the enzymes that it can potentially produce. Enzymes are like a blueprint for the reactions they can catalyze. (B) Using the enzymes we identify the reactions in the organism. (C) We construct the stoichiometric matrix of the metabolic model. (D) We consider the flux space under different conditions (e.g., steady states); they correspond to polytopes containing flux vectors addressing these conditions. (E) We sample from polytopes that are typically skinny and of high dimension. (F) The distribution of the flux of a reaction provides great insights to biologists.

community. This will allow us to further investigate the dynamics, the functional profile, and the inter-species reactions that occur. Using the stoichiometry of each reaction, which is always the same in the various species, we convert the metabolic network of an organism to a mathematical model. Thus, the metabolic network becomes an *in silico* model of the knowledge it represents. In metabolic networks analysis mass and energy are considered to be conserved [46]. As many homeostatic states, that is steady internal conditions [54], are close to steady states (where the production rate of each metabolite equals its consumption rate [8]) we commonly use the latter in metabolic networks analysis.

Stoichiometric coefficients are the number of molecules a biochemical reaction consumes and produces. The coefficients of all the reactions in a network, with m metabolites and n reactions ($m < n$), form the *stoichiometric matrix* $S \in \mathbb{R}^{m \times n}$ [47]. The nullspace of S corresponds to the steady states of the network:

$$S \cdot x = 0, \tag{1}$$

where $x \in \mathbb{R}^n$ is the *flux vector* that contains the fluxes of each chemical reaction of the network. Flux is the rate of turnover of molecules through a metabolic pathway.

All physical variables are finite, therefore the flux (and the concentration) is bounded [47]; that is for each coordinate x_i of the x, there are $2n$ constants $x_{ub,i}$ and $x_{lb,i}$ such that $x_{lb,i} \leq x_i \leq x_{ub,i}$, for $i \in [n]$. We derive the constraints from explicit experimental information. In cases where there is no such information, reactions are left unconstrained by setting arbitrary large values to their corresponding bounds according to their reversibility properties; i.e., if a reaction is reversible then its flux might be negative as well [38]. The constraints define a n-dimensional box containing both the steady and the dynamic states of the system. If we intersect that box with the nullspace of S, then we define a polytope

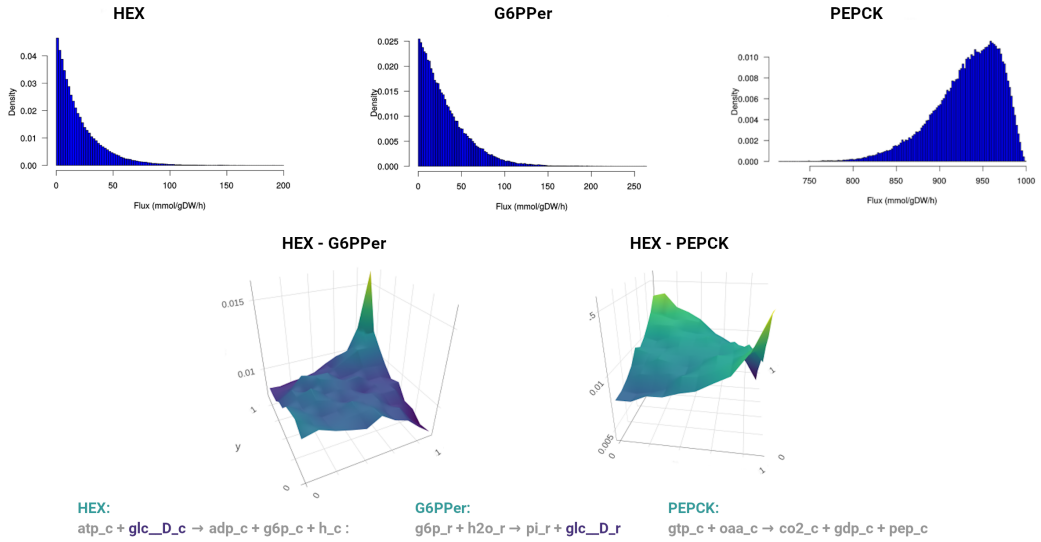


Figure 2 Flux distributions in the most recent human metabolic network Recon3D [7]. We estimate the flux distributions of the reactions catalyzed by the enzymes Hexokinase (D-Glucose:ATP) (HEX), Glucose-6-Phosphate Phosphatase, Edoplasmic Reticular (G6PPer) and Phosphoenolpyruvate carboxykinase (GTP) (PEPCK). As we sample steady states, the production rate of *glc___D_c* should be equal to its consumption rate. Thus, in the corresponding copula, we see a positive dependency between HEX, i.e., the reaction that consumes *glc___D_c* and G6PPer, that produces it. Furthermore, the PEPCK reaction operates when there is no *glc___D_c* available and does not operate when the latter is present. Thus, in their copula we observe a negative dependency between HEX and PEPCK. A copula is a bivariate probability distribution for which the marginal probability distribution of each variable is uniform. It implies a positive dependency when the mass of the distribution concentrates along the up-diagonal (HEX - G6PPer) and a negative dependency when the mass is concentrated along the down-diagonal (HEX - PEPCK). The bottom line contains the reactions and their stoichiometry.

that encodes all the possible steady states and their flux distributions [47]. We call it the steady-state *flux space*. Fig. 1 illustrates the complete workflow from building a metabolic network to the computation of a flux distribution.

Using the polytopal representation, a commonly used method for the analysis of a metabolic network is Flux Balance Analysis (FBA) [45]. FBA identifies a single optimal flux distribution by optimizing a linear objective function over a polytope [45]. Unfortunately, this is a *biased* method because it depends on the selection of the objective function. To study the global features of a metabolic network we need *unbiased methods*. To obtain an accurate picture of the whole solution space we exploit sampling techniques [52]. If collect a sufficient number of points uniformly distributed in the interior of the polytope, then the biologists can study the properties of certain components of the whole network and deduce significant biological insights [47]. Therefore, efficient sampling tools are of great importance.

1.3 Metabolic networks through the lens of random sampling

Efficient uniform random sampling on polytopes resulting from metabolic networks is a very challenging task both from the theoretical (algorithmic) and the engineering (implementation) point of view. First, the dimension of the polytopes is of the order of certain thousands. This requires, for example, advanced engineering techniques to cope with memory requirements and to perform linear algebra operations with large matrices; e.g., in Recon3D [7] we compute

the null space of a $8\,399 \times 13\,543$ matrix. Second, the polytopes are rather skinny (Sec. 4); this makes it harder for sampling algorithms to move in the interior of polytopes and calls for novel practical techniques to sample.

There is extended on-going research concerning advanced algorithms and implementations for sampling metabolic networks over the last decades. Markov Chain Monte Carlo algorithms such as Hit-and-Run (HR) [55] have been widely used to address the challenges of sampling. Two variants of HR are the non-Markovian Artificial Centering Hit-and-Run (ACHR) [30] that has been widely used in sampling metabolic models, e.g., [51], and Coordinate Hit-and-Run with Rounding (CHRR) [25]. The latter is part of the `cobra` toolbox [26], the most commonly used software package for the analysis of metabolic networks. CHRR enables sampling from complex metabolic networks corresponding to the highest dimensional polytopes so far. There are also stochastic formulations where the inclusion of experimental noise in the model makes it more compatible with the stochastic nature of biological networks [39]. The recent study in [19] offers an overview as well as an experimental comparison of the currently available samplers.

These implementations played a crucial role in actually performing in practice uniform sampling from the flux space. However, they are currently limited to handle polytopes of dimension say $\leq 2\,500$ [19, 25]. This is also the order of magnitude of the most complicated, so far, metabolic network model built, Recon3D [7]. By including 13 543 metabolic reactions and involving 4 140 unique metabolites, Recon3D provides a representation of the 17% of the functionally of annotated human genes. To our knowledge, there is no method that can efficiently handle sampling from the flux space of Recon3D.

Apparently, the dimension of the polytopes will keep rising and not only for the ones corresponding to human metabolic networks. Metabolism governs systems biology at all its levels, including the one of the community. Thus, we are not only interested in sampling a sole metabolic network, even if it has the challenges of the human. Sampling in polytopes associated to network of networks are the next big thing in metabolic networks analysis and in Systems Biology [4, 48].

Regarding the sampling process, from the theoretical point of view, we are interested in the convergence time, or *mixing time*, of the Markov Chain, or geometric *random walk*, to the target distribution. Given a d-dimensional polytope P, the mixing time of several geometric random walks (e.g., HR or Ball Walk) grows quadratically with respect to the sandwiching ratio R/r of the polytope [36, 37]. Here r and R are the radii of the smallest and largest ball with center the origin that contains, and is contained, in P, respectively; i.e., $rB_d \subseteq P \subseteq RB_d$, where B_d is the unit ball. It is crucial to reduce R/r, i.e., to put P in well a rounded position where $R/r = \widetilde{\mathcal{O}}(\sqrt{d})$; the $\widetilde{\mathcal{O}}(\cdot)$ notation means that we are ignoring polylogarithmic factors. A powerful approach to obtain well roundness is to put P in *near isotropic position*. In general, $K \subset \mathbb{R}^d$ is in isotropic position if the uniform distribution over K is in isotropic position, that is $\mathbb{E}_{X\sim K}[X] = 0$ and $\mathbb{E}_{X\sim K}[X^T X] = I_d$, where I_d is the $d \times d$ identity matrix. Thus, to put a polytope P into isotropic position one has to generate a set of uniform points in its interior and apply to P the transformation that maps the point-set to isotropic position; then iterate this procedure until P is in c-isotropic position [17, 37], for a constant c. In [1] they prove that $\mathcal{O}(d)$ points suffice to achieve 2-isotropic position. Alternatively in [25] they compute the maximum volume ellipsoid in P, they map it to the unit ball, and then apply to P the same transformation. They experimentally show that a few iterations suffice to put P in John's position [28]. Moreover, there are a few algorithmic contributions that combine sampling with distribution isotropization steps, e.g., the multi-point walk [5] and the annealing schedule [29].

An important parameter of a random walk is the *walk length*, i.e., the number of the intermediate points that a random walk visits before producing a single sample point. The longer the walk length of a random walk is, the smaller the distance of the current distribution to the stationary (target) distribution becomes. For the majority of random walks there are bounds on the walk length to bound the mixing time with respect to a statistical distance. For example, HR generates a sample from a distribution with total variation distance less than ϵ from the target distribution after $\widetilde{\mathcal{O}}(d^3)$ [37] steps, in a well rounded convex body and for log-concave distributions. Similarly, CDHR mixes after a polynomial, in the diameter and the dimension, number of steps [34, 42] for the case of uniform distribution. However, extended practical results have shown that both CDHR and HR converges after $\mathcal{O}(d^2)$ steps [10, 17, 25]. The leading algorithms for uniform polytope sampling are the Riemannian Hamiltonian Monte Carlo sampler [35] and the Vaidya walk [14], with mixing times $\widetilde{\mathcal{O}}(md^{2/3})$ and $\widetilde{\mathcal{O}}(m^{1/2}d^{3/2})$ steps, respectively. However, it is not clear if these random walks can outperform CDHR in practice, because of their high cost per step and numerical instability.

Billiard Walk (BW) [23] is a random walk that employs linear trajectories in a convex body with boundary reflections; alas with an unknown mixing time. The closest guarantees for its mixing time are those of HR and stochastic billiards [18]. Interestingly, [23] shows that, experimentally, BW converges faster than HR for a proper tuning of its parameters. The same conclusion follows from the computation of the volume of zonotopes [11]. It is not known how the sandwiching ratio of P affects the mixing time of BW. Since BW employs reflections on the boundary, we can consider it as a special case of Reflective Hamiltonian Monte Carlo [15].

For almost all random walks the theoretical bounds on their mixing times are pessimistic and unrealistic for computations. Hence, if we terminate the random walk earlier, we generate samples that are usually highly correlated. There are several *MCMC Convergence Diagnostics* [50] to check if the quality of a sample can provide an accurate approximation of the target distribution. For a dependent sample, a powerful diagnostic is the *Effective Sample Size* (ESS). It is the number of effectively independent draws from the target distribution that the Markov chain is equivalent to. For autocorrelated samples, ESS bounds the uncertainty in estimates [21] and provides information about the quality of the sample. There are several statistical tests to evaluate the quality of a generated sample, e.g., potential scale reduction factor (PSRF) [20], maximum mean discrepancy (MMD) [22], and the uniform tests [16]. Interestingly, the copula representation we employ in Fig. 2 to capture the dependence between two fluxes of reactions was also used successfully in a geometric framework to detect financial crises capturing the dependence between portfolio return and volatility [9].

1.4 Our contribution

We introduce a Multi-phase Monte Carlo Sampling (MMCS) algorithm (Sec. 3 and Alg. 1) to sample from a polytope P. In particular, we split the sampling procedure in phases where, starting from P, each phase uses the sample to round the polytope. This improves the efficiency of the random walk in the next phase, see Fig. 3. For sampling, we propose an improved variant of Billiard Walk (BW) (Sec. 2 that enjoys faster arithmetic complexity per step. We also handle efficiently the potential arithmetic inaccuracies near to the boundary, see [15]. We accompany the MMCS algorithm with a powerful MCMC diagnostic, namely the estimation of Effective Sample Size (ESS), to identify a satisfactory convergence to the uniform distribution. However, our method is flexible and we can use any random walk and combination of MCMC diagnostics to decide convergence.

The open-source implementation of our algorithms[1] provides a complete software framework to handle efficiently sampling in metabolic networks. We demonstrate the efficiency of our tools by performing experiments on almost all the metabolic networks that are publicly available and by comparing with the state-of-the-art software packages as `cobra` (Sec. 4.2). Our implementation is faster than `cobra` for low dimensional models, with a speed-up that ranges from 10 to 100 times; this gap on running times increases for bigger models (Table 1). The quality of the sample our software produces is measured with two widely used diagnostics, i.e., ESS and potential scale reduction factor (PSRF) [20]. The highlight of our method is the ability to sample from the most complicated human metabolic network that is accessible today, namely Recon3D. In Fig. 2 we estimate marginal univariate and bivariate flux distributions in Recon3D which validate (a) the quality of the sample by confirming a mutually exclusive pair of biochemical pathways, and that (b) our method indeed generates steady states. In particular, our software can sample $1.44 \cdot 10^5$ points from a 5 335-dimensional polytope in a day using modest hardware. This set of points suffices for the majority of systems biology analytics. To our understanding this task is out of reach for existing software. Last, MMCS algorithm is quite general sampling scheme and so it has the potential to address other hard computational problems like multivariate integration and volume estimation of polytopes.

2 Efficient Billiard walk

The geometric random walk of our choice to sample from a polytope is based on Billiard Walk (BW) [23], which we modify to reduce the per-step cost.

For a polytope $P = \{x \in \mathbb{R}^d \,|\, Ax \leq b\}$, where $A \in \mathbb{R}^{k \times d}$ and $b \in \mathbb{R}^k$, BW starts from a given point $p_0 \in P$, selects uniformly at random a direction, say v_0, and it moves along the direction of v_0 for length L; it reflects on the boundary if necessary. This results a new point p_1 inside P. We repeat the procedure from p_1. Asymptotically it converges to the uniform distribution over P. The length is $L = -\tau \ln \eta$, where η is a uniform number in $(0, 1)$, that is $\eta \sim \mathcal{U}(0, 1)$, and τ is a predefined constant. It is useful to set a bound, say ρ, on the number of reflections to avoid computationally hard cases where the trajectory may stuck in corners. In [23] they set $\tau \approx diam(P)$ and $\rho = 10d$. Our choices for τ and ρ depend on a burn-in step that we detail in Sec. 4.

At each step of BW we compute the intersection point of a ray, say $\ell := \{p + tv,\ t \in \mathbb{R}_+\}$, with the boundary of P, ∂P, and the normal vector of the tangent plane at the intersection point. The inner vector of the facet that the intersection point belongs to is a row of A. To compute the point $\partial P \cap \ell$ where the first reflection of a BW step takes place, we solve the following m linear equations

$$a_j^T(p_0 + t_j v_0) = b_j \Rightarrow t_j = (b_j - a_j^T p_0)/a_j^T v_0,\ j \in [k], \tag{2}$$

and keep the smallest positive t_j; a_j is the j-th row of the matrix A. We solve each equation in $\mathcal{O}(d)$ operations and so the overall complexity is $\mathcal{O}(dk)$. A straightforward approach for BW would consider that each reflection costs $\mathcal{O}(kd)$ and thus the per step cost is $\mathcal{O}(\rho kd)$. However, our improved version performs more efficiently both *point* and *direction updates* by storing computations from the previous iteration combined with a preprocessing step. The preprocessing step involves the normal vectors of the facets, that takes $m^2 d$ operations, and the amortized per-step complexity of BW becomes $\mathcal{O}((\rho + d)k)$.

[1] `https://github.com/GeomScale/volume_approximation/tree/socg21`

▶ **Lemma 1.** *The amortized per step complexity of BW is $\mathcal{O}((\rho + d)k)$ after a preprocessing step that takes $\mathcal{O}(k^2 d)$ operations, where ρ is the maximum number of reflections per step.*

The use of floating point arithmetic could result to points outside P due to rounding errors when computing boundary points. To avoid this, when we compute the roots in Equation (2) we exclude the facet that the ray hit in the previous reflection.

3 Multiphase Monte Carlo Sampling algorithm

To sample steady states in the flux space of a metabolic network, with m metabolites and n reactions, we introduce a Multiphase Monte Carlo Sampling (MMCS) algorithm; it is multiphase because it consists of a sequence of sampling phases.

Let $S \in \mathbb{R}^{m \times n}$ be the stoichiometric matrix and $x_{lb},\ x_{ub} \in \mathbb{R}^n$ bounds on the fluxes. The flux space is the bounded convex polytope

$$\text{FS} := \{x \in \mathbb{R}^n \,|\, Sx = 0,\ x_{lb} \leq x \leq x_{ub}\} \subset \mathbb{R}^n. \tag{3}$$

The dimension, d, of FS is smaller than the dimension of the ambient space; that is $d \leq n$. To work with a full dimensional polytope we restrict the box induced by the inequalities $x_{lb} \leq x \leq x_{ub}$ to the null space of S. Let the H-representation of the box be $\left\{x \in \mathbb{R}^n \,\middle|\, \begin{pmatrix} I_n \\ -I_n \end{pmatrix} x \leq \begin{pmatrix} x_{ub} \\ x_{lb} \end{pmatrix}\right\}$, where I_n is the $n \times n$ identity matrix, and let $N \in \mathbb{R}^{n \times d}$ be the matrix of the null space of S, that is $S\,N = 0_{m \times d}$. Then $P = \{x \in \mathbb{R}^d \mid Ax \leq b\}$, where $A = \begin{pmatrix} I_n N \\ -I_n N \end{pmatrix}$ and $b = \begin{pmatrix} x_{ub} \\ x_{lb} \end{pmatrix} N$, is a full dimensional polytope (in $\mathbb{R}^d$). After we sample (uniformly) points from P, we transform them to uniformly distributed points (that is steady states) in FS by applying the linear map induced by N.

MMCS generates, in a sequence of sampling phases, a set of points, that is almost equivalent to n independent uniformly distributed points in P, where n is given. At each phase, it employs Billiard Walk (Section 2) to sample approximate uniformly distributed points, rounding to speedup sampling, and uses the Effective Sample Size (ESS) diagnostic to decide termination. The pseudo-code of the algorithm appears in Alg. 1.

Overview. Initially we set $P_0 = P$.

At each phase $i \geq 0$ we sample at most λ points from P_i. We generate them in chunks; we also call them *chain* of sampling points. Each chain contains at most l points (for simplicity consider $l = \mathcal{O}(1)$). To generate the points in each chain we employ BW, starting from a point inside P_i; the starting point is different for each chain. We repeat this procedure until the total number of samples in P_i reaches the maximum number λ; we need $\frac{\lambda}{l}$ chains. To compute a starting point for a chain, we pick a point uniformly at random in the Chebychev ball of P_i and we perform $\mathcal{O}(\sqrt{d})$ burn-in BW steps to obtain a warm start.

After we have generated λ sample points we perform a rounding step on P_i to obtain the polytope of the next phase, P_{i+1}. We compute a linear transformation, T_i, that puts the sample into isotropic position and then $P_{i+1} = T_i(P_i)$. The efficiency of BW improves from one phase to the next one because the sandwiching ratio decreases and so the average number of reflections decreases and thus the convergence to the uniform distribution accelerates (Section 4.2). That is we obtain faster a sample of better quality. Finally, the (product of the) inverse transformations maps the samples to $P_0 = P$. Fig. 3 depicts the procedure.

Termination. There are no bounds on the mixing time of BW [23], hence for termination we rely on ESS. MMCS terminates when the minimum ESS among all the univariate marginals is larger than a requested value. We chose the marginal distributions (of each flux) because

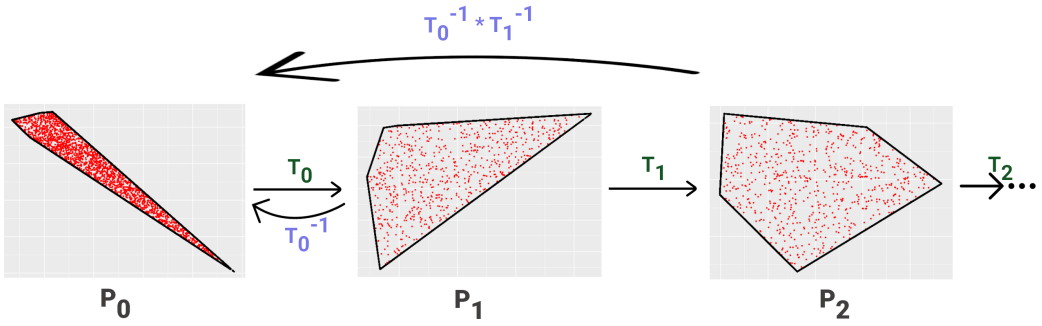


Figure 3 An illustration of our Multiphase Monte Carlo Sampling algorithm. The method is given an integer n and starts at phase $i = 0$ sampling from P_0. In each phase it samples a maximum number of points λ. If the sum of Effective Sample Size in each phase becomes larger than n before the total number of samples in P_i reaches λ then the algorithm terminates. Otherwise, we proceed to a new phase. We map back to P_0 all the generated samples of each phase.

they are essential for systems biologists, see [6] for a typical example. In particular, after we generate a chain, the algorithm updates the ESS of each univariate marginal to take into account all the points that we have sampled in P_i, including the newly generated chain. We keep the minimum, say n_i, among all marginal ESS values. If $\sum_{j=0}^{i} n_j$ becomes larger than n before the total number of samples in P_i reaches the upper bound λ, then MMCS terminates. Otherwise, we proceed to the next phase. In summary, MMCS terminates when the sum of the minimum marginal ESS values of each phase reaches n.

Rounding step. This step is motivated by the theoretical result in [1] and the rounding algorithms [37, 17]. We apply the linear transformation T_i to P_i so that the sandwiching ratio of P_{i+1} is smaller than that of P_i. To find the suitable T_i we compute the SVD decomposition of the matrix that contains the sample row-wise [3].

Updating the Effective Sample Size. The effective sample size of a sample of points generated by a process with autocorrelations ρ_t at lag t is function (actually an infinite series) in the ρ_t's; its exact value is unknown. Following [21], we efficiently compute ESS employing a finite sum of monotone estimators $\hat{\rho}_t$ of the autocorrelation at lag t, by exploiting Fast Fourier Transform. Furthermore, given M chains of samples, the autocorrelation estimator $\hat{\rho}_t$ is given by, $\hat{\rho}_t = 1 - \frac{C - \frac{1}{M}\sum_{i=1}^{M} \hat{\rho}_{t,i}}{B}$, where C and B are the within-sample variance estimate and the multi-chain variance estimate given in [20] and $\hat{\rho}_{t,i}$ is an estimator of the autocorrelation of the i-th chain at lag t. To update the ESS, for every new chain of points the algorithm generates, we compute $\hat{\rho}_{t,i}$. Then, using Welford's algorithm we update the average of the estimators of autocorrelation at lag t, as well as the between-chain variance and the within-sample variance estimators given in [20]. Finally, we update the ESS using these estimators.

▶ **Lemma 2.** *Let $P = \{x \in \mathbb{R}^d \mid Ax \leq b\}$, $A \in \mathbb{R}^{k \times d}$, $b \in \mathbb{R}^k$ a full dimensional polytope in $\mathbb{R}^d$. The total number of operations per phase that Alg. 1 performs, is $\mathcal{O}(W(\rho+d)k\lambda + \lambda^2 d + d^3)$, where W is the walk length for Billiard Walk.*

In Section 4 we discuss how to tune the parameters of MMCS to make it more efficient in practice. We also comment on the (practical) complexity of each phase, based on the tuning.

Algorithm 1 Multiphase Monte Carlo Sampling$(P, n, l, \lambda, \rho, \tau, W)$.

Input : A full dimensional polytope $P \in \mathbb{R}^d$; Requested effectiveness $n \in \mathbb{N}$; l length of each chain; upper bound for the number of generated points in each phase λ; upper bound on the number of reflections ρ; length of trajectory parameter τ; walk length W.

Output : A set of approximate uniformly distributed points $S \in P$

Set $P_0 \leftarrow P,\ sum\_ess \leftarrow 0,\ S \leftarrow \emptyset,\ i \leftarrow 0,\ T_0 = I_d$;
do
  $sum\_point\_phase \leftarrow 0,\ U \leftarrow \emptyset$;
  do
    Generate a starting point $p \in P_i$;
    Generate a set Q of l points with Billiard Walk starting from p;
    $S \leftarrow S \cup T_i^{-1}(Q)$; $U \leftarrow U \cup Q$;
    $sum\_point\_phase \leftarrow sum\_point\_phase + l$;
    Update ESS n_i of this phase;
    if $sum\_ess + n_i \geq n$ **then break** ;
  while $sum\_point\_phase < \lambda$;
  $sum\_ess \leftarrow sum\_ess + n_i$; $i \leftarrow i + 1$;
  Compute T such that $T(U)$ is in isotropic position;
  $T_i \leftarrow T_{i-1} \circ T$;
while $sum\_ess < n$;
return S;

4 Implementation and Experiments

In the sequel we present the implementation of our approach and the tuning of various parameters. We present experiments in an extended set of BURG models [31], including the most complex metabolic networks i.e., the human Recon2D [56] and Recon3D [7]. We end up to sample from polytopes of thousands of dimensions and show that our method can estimate precisely the flux distributions. We analyze various aspects of our method as the runtime, the efficiency and the quality of the output. We compare our method against the state-of-the-art software for the analysis of metabolic networks, which is the `Matlab toolbox` of `cobra` [26]. Our implementation for low dimensional networks is two orders of magnitude faster than `cobra`. As the dimension grows this gap on the run-time increases. The fast mixing of billiard walk allow us to use all the generated samples to approximate each flux distribution improving the flux distribution estimation.

We provide a complete open-source software framework to handle big metabolic networks. The framework loads a metabolic model in some standard format (e.g., `mat, json` files) and performs an analysis of the model e.g., compute the marginal distributions of a given metabolite. All the results in this paper are reproducible using our publicly available code[2]. The core of our implementation is in `C++` to optimize performance while the user interface is implemented in `R`. The package employs `eigen` [24] for linear algebra, `boost` [41] for random number generation, `mosek` [2] as the linear program solver, and expands `volesti` [12] an open-source package for high dimensional sampling and volume approximation. All experiments were performed on a PC with `Intel® Core i7-6700 3.40GHz × 8 CPU` and `32GB RAM`.

[2] `https://github.com/GeomScale/volume_approximation/tree/socg21`

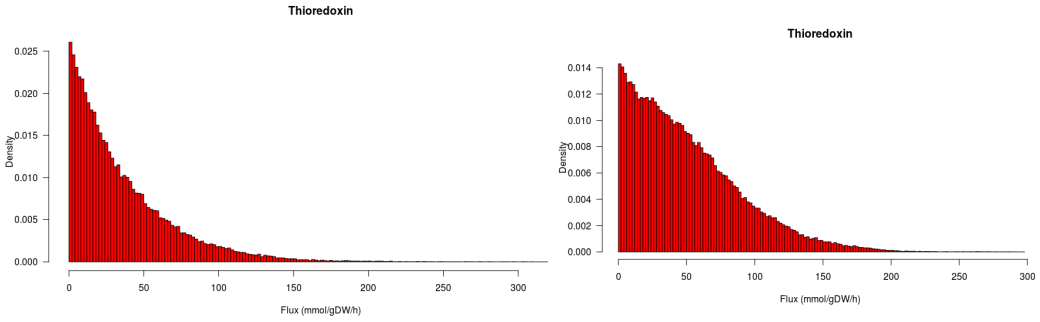


Figure 4 Our method estimation of the marginal distribution of the "Thioredoxin reductace" flux in a constraint-based model of Homo Sapiens metabolism Recon2D [56] (left) and Recon3D [7] (right).

4.1 Parameter tuning for practical performance

We give details on how we tune various parameters presented in Section 3 in our implementation.

Parameters of Billiard Walk. To employ Billiard Walk (Section 2) we have to make efficient selections for the parameter τ that controls the length of the trajectory in each step, for the maximum number of reflections per step ρ, and for the walk length W of the random walk. We experimentally found that setting $W = 1$ the empirical distribution converges faster to the uniform distribution. Thus, we get a higher ESS faster than the case of $W > 1$. To set τ in phase i, first we set $\tau = 6\sqrt{d}r$ where r is the radius of the Chebychev ball of P_i. Then, we start from the center of the Chebychev ball, we perform $100 + 4\sqrt{d}$ Billiard Walk steps and we store all the points in a set Q. Then we set $\tau = \max\{\max_{q \in Q}\{||q - p||_2\}, 6\sqrt{d}r\}$. For the maximum number of reflections we found experimentally that $\rho = 100d$ is violated in less than 0.1% of the total number of Billiard Walk steps in our experiments.

Rounding step. In each phase i of our method, when the minimum value of ESS among all the marginals has not reached the requested threshold, we use the generated sample to perform a rounding step by mapping the points to isotropic position. After computing the SVD decomposition of the point-set we also rescale the singular values such that the smallest one is 1, to improve numerical stability as suggested in [17]. We found experimentally that setting the maximum number of Billiard Walk points per phase $\lambda = 20d$, where d is the dimension of the polytope, suffice to improve the roundness from phase to phase. When, in any phase, the ratio between the maximum over the minimum singular value is smallest than 3 we stop performing any new rounding step. In that case we stay on the current phase until we reach the requested value of ESS.

▶ Remark 1. Given the Stoichiometric matrix $S \in \mathbb{R}^{m \times n}$ of a metabolic network with flux bounds $x_{lb} \leq x \leq x_{ub}$, the total number of operations per phase that our implementation of Alg. 1 performs, according the parameterization given in this Section is $\mathcal{O}(nd^2)$, where d is the dimension of the null space of S and n is the number of reactions occur in the metabolic network.

4.2 Experiments

We test and evaluate our software on 17 models from the BIGG database [31] and Recon2D, Recon3D from [44]. In particular, we sample from models that correspond to polytopes of dimension less than 100; the simplest model in this setting is the well known bacteria *Escherichia Coli*. We also computed with models that correspond to polytopes of dimension a few thousands; this is the case for Recon2D and Recon3D. We do not employ parallelism for any implementation, thus we report only sequential running times. To assess the quality of our results we employ a second MCMC convergence diagnostic besides the Effective Sample Size (ESS). This is the potential scale reduction factor (PSRF), introduced by Rubin and Gelman [20]. In particular, we compute the PSRF for each univariate marginal of the sample that MMCS outputs. Following [20], a convergence is satisfying according to PSRF when all the marginals have PSRF smaller than 1.1.

The workflow of `cobra` for sampling first performs a rounding step and then samples using Coordinate Directions Hit-and-Run (CDHR). To compare with `cobra` we set the walk length of CDHR according to the empirical suggestion made in [25], i.e., equal to $8d^2$, where d is the dimension of the polytope we sample. For Recon2D we follow the paradigm in [25] which shows that the method converges for walk length equal to $1.57e{+}08$. To have a fair comparison we let `cobra` to sample a minimum number of 1 000 points. If in the computed sample there is a marginal with PSRF larger than 1.1 we continue sampling until all PSRFs are smaller than 1.1.

Table 1 17 metabolic networks from [31] and Recon2D, Recon3D from [44]; (m) the number of Metabolites, (n) the number of Reactions, (d) the dimension of the polytope; (N) is the total number of sampled points × walk length; for MMCS we stop when the sum of the minimum value of ESS among all the univariate marginals in each phase is 1000 (we report the number of phases in parenthesis); for `cobra` we set the walk length to $8d^2$ and $1.57e{+}08$ for Recon2D following [25], sample at least 1000 points and stop when all marginals have PSRF < 1.1; the runtime of `cobra` for Recon2D is an estimation of the sequential time just for the purpose of the comparison in this paper.

				MMCS		`cobra`	
name	(m)	(n)	(d)	Time (sec)	(N)	Time (sec)	(N)
e_coli_core	72	95	24	6.50e-01	3.40e+03 (8)	7.20e+01	4.61e+06
iLJ478	570	652	59	9.00e+00	5.40e+03 (5)	4.54e+02	2.79e+07
iSB619	655	743	83	1.70e+01	8.20e+03 (5)	9.56e+02	5.51e+07
iHN637	698	785	88	2.00e+01	6.80e+03 (4)	1.03e+03	6.19e+07
iJN678	795	863	91	2.50e+01	8.10e+03 (4)	1.17e+03	6.62e+07
iNF517	650	754	92	1.70e+01	6.20e+03 (4)	1.33e+03	6.77e+07
iJN746	907	1054	116	5.70e+01	8.70e+03 (3)	2.22e+03	1.07e+08
iAB_RBC_283	342	469	130	5.20e+01	1.07e+04 (5)	7.85e+03	4.05e+08
iJR904	761	1075	227	2.98e+02	1.62e+04 (4)	8.81e+03	4.12e+08
iAT_PLT_636	738	1008	289	3.25e+02	1.04e+04 (2)	1.73e+04	6.68e+08
iSDY_1059	1888	2539	509	2.813e+03	2.31e+04 (3)	6.66e+04	2.07e+09
iAF1260	1668	2382	516	6.84e+03	5.33e+04 (6)	7.04e+04	2.13e+09
iEC1344_C	1934	2726	578	4.86e+03	3.95e+04 (4)	9.42e+04	2.67e+09
iJO1366	1805	2583	582	6.02e+03	5.14e+04 (5)	9.99e+04	2.71e+09
iBWG_1329	1949	2741	609	3.06e+03	4.22e+04 (4)	1.05e+05	2.97e+09
iML1515	1877	2712	633	4.65e+03	5.65e+04 (5)	1.15e+05	3.21e+09
Recon1	2766	3741	931	8.09e+03	1.94e+04 (2)	3.20e+05	6.93e+09
Recon2D	5063	7440	2430	2.48e+04	5.44e+04 (2)	$\sim$ 140 days	$1.57e{+}11$
Recon3D	8399	13543	5335	1.03e+05	1.44e+05 (2)	–	–

In Table 1 we report the results of MMCS and `cobra`. We run MMCS until we get a value of ESS equal to 1 000; meaning that we stop when the sum over all phases of the minimum values of ESS among all the marginals is larger than 1 000. Moreover, in Table 1 all the marginals of the sample that MMCS returns have PSRF < 1.1. This is another statistical evidence on the quality of the generated sample. The histograms in Fig. 4 illustrate an approximation for the flux distribution of the reaction Thioredoxin as computed in Recon2D and Recon3D respectively. The same marginal flux distribution in Recon2D was estimated also in [25]. Notice that the estimated density slightly changes in Recon3D as the stoichiometric matrix has been updated and thus the corresponding marginal is affected. In Fig. 2 we also employ the copula representation to capture the dependency between two fluxes to confirm a mutually exclusive pair of biochemical pathways. Notice that the run-time of MMCS is one or two orders of magnitude smaller than the run-time of `cobra` and this gap becomes much larger for higher dimensional models such as Recon2D and Recon3D.

For some models –see in the full version of the paper [13]– we introduce a further improvement to obtain a better convergence. If there is a marginal in the generated sample from MMCS that has a PSRF larger than 1.1 then we do not take into account the k first phases, starting with $k = 1$ until we get both ESS equal to 1 000 and all the PSRF values smaller than 1.1 for all the marginals. By "do not take into account" we mean that we neither store the generated sample –for the first k phases– nor we sum up its ESS to the overall ESS considered for termination by MMCS. Note that for these models it is not practical to repeat MMCS runs for different k until we get the required PSRF value. We can obtain the final results –reported in Tables 1– in one pass. We simply drop a phase when the ESS reaches the requested value but the PSRF is not smaller than 1.1 for all the marginals.

Interestingly, the total number of Billiard Walk steps –and consequently the run-time– does not increase as k increases [13]. This means that the performance of our method improves for these models, when we do not take into account the k first phases of MMCS. This happens because the performance of Billiard Walk improves as the polytope becomes more rounded from phase to phase. In particular, in the full version of the paper [13] we analyze the performance of Billiard Walk for the model iAF1260. We sample $20d$ points per phase with walk length equal to 1 and we report the average number of reflections, the ESS, the run-time, and the ratio $\sigma_{\max}/\sigma_{\min}$ per phase. The latter is the ratio between the maximum over the minimum singular value of the point-set. The larger this ratio is the more skinny the polytope of the corresponding phase is. As the method progresses from the first to the last phase, the average number of reflections and the run-time decrease and the ESS increases. This means that as the polytope becomes more rounded from phase to phase, the Billiard Walk step becomes faster and the generated sample has better quality. This explains why the total run-time does not increase when we do not take into account the first k phases: the initial phases are slow and they contribute poorly to the quality of the final sample; the last phases are fast and contribute with more accurate samples.

References

1 Radosław Adamczak, Alexander Litvak, Alain Pajor, and Nicole Tomczak-Jaegermann. Quantitative estimates of the convergence of the empirical covariance matrix in log-concave ensembles. *Journal of the American Mathematical Society*, 23(2):535–561, 2010. doi:10.1090/S0894-0347-09-00650-X.

2 MOSEK ApS. *The MOSEK optimization toolbox for R manual. Version 9.2*, 2019. URL: https://docs.mosek.com/9.2/rmosek/index.html.

3 Shiri Artstein-Avidan, Haim Kaplan, and Micha Sharir. On radial isotropic position: Theory and algorithms, 2020. arXiv:2005.04918.

4 David B Bernstein, Floyd E Dewhirst, and Daniel Segre. Metabolic network percolation quantifies biosynthetic capabilities across the human oral microbiome. *Elife*, 8:e39733, 2019.

5 Dimitris Bertsimas and Santosh Vempala. Solving convex programs by random walks. *J. ACM*, 51(4):540–556, 2004. doi:10.1145/1008731.1008733.

6 Sergio Bordel, Rasmus Agren, and Jens Nielsen. Sampling the solution space in genome-scale metabolic networks reveals transcriptional regulation in key enzymes. *PLOS Computational Biology*, 6(7):1–13, July 2010. doi:10.1371/journal.pcbi.1000859.

7 Elizabeth Brunk, Swagatika Sahoo, Daniel C Zielinski, Ali Altunkaya, Andreas Dräger, Nathan Mih, Francesco Gatto, Avlant Nilsson, German Andres Preciat Gonzalez, Maike Kathrin Aurich, et al. Recon3D enables a three-dimensional view of gene variation in human metabolism. *Nature biotechnology*, 36(3):272, 2018.

8 Ali Cakmak, Xinjian Qi, A Ercument Cicek, Ilya Bederman, Leigh Henderson, Mitchell Drumm, and Gultekin Ozsoyoglu. A new metabolomics analysis technique: steady-state metabolic network dynamics analysis. *Journal of bioinformatics and computational biology*, 10(01):1240003, 2012.

9 Ludovic Calès, Apostolos Chalkis, Ioannis Z. Emiris, and Vissarion Fisikopoulos. Practical Volume Computation of Structured Convex Bodies, and an Application to Modeling Portfolio Dependencies and Financial Crises. In Bettina Speckmann and Csaba D. Tóth, editors, *34th International Symposium on Computational Geometry (SoCG 2018)*, volume 99 of *LIPIcs*, pages 19:1–19:15, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.SoCG.2018.19.

10 Apostolos Chalkis, Ioannis Z. Emiris, and Vissarion Fisikopoulos. Practical volume estimation by a new annealing schedule for cooling convex bodies, 2019. arXiv:1905.05494.

11 Apostolos Chalkis, Ioannis Z. Emiris, and Vissarion Fisikopoulos. Practical volume estimation of zonotopes by a new annealing schedule for cooling convex bodies. In Anna Maria Bigatti, Jacques Carette, James H. Davenport, Michael Joswig, and Timo de Wolff, editors, *Mathematical Software – ICMS 2020*, pages 212–221, Cham, 2020. Springer International Publishing.

12 Apostolos Chalkis and Vissarion Fisikopoulos. volesti: Volume approximation and sampling for convex polytopes in R, 2020. arXiv:2007.01578.

13 Apostolos Chalkis, Vissarion Fisikopoulos, Elias Tsigaridas, and Haris Zafeiropoulos. Geometric algorithms for sampling the flux space of metabolic networks, 2021. arXiv:2012.05503.

14 Yuansi Chen, Raaz Dwivedi, Martin J. Wainwright, and Bin Yu. Fast mcmc sampling algorithms on polytopes. *Journal of Machine Learning Research*, 19(55):1–86, 2018. URL: http://jmlr.org/papers/v19/18-158.html.

15 A. Chevallier, S. Pion, and F. Cazals. Hamiltonian Monte Carlo with boundary reflections, and application to polytope volume calculations. Research Report RR-9222, INRIA Sophia Antipolis, France, 2018. URL: https://hal.archives-ouvertes.fr/hal-01919855.

16 B. Cousins. *Efficient high-dimensional sampling and integration.* PhD thesis, Georgia Institute of Technology, Georgia, U.S.A., 2017.

17 Ben Cousins and Santosh Vempala. A practical volume algorithm. *Mathematical Programming Computation*, 8(2):133–160, 2016.

18 A. B. Dieker and Santosh S. Vempala. Stochastic billiards for sampling from the boundary of a convex set. *Mathematics of Operations Research*, 40(4):888–901, 2015. URL: http://www.jstor.org/stable/24540983.

19 Shirin Fallahi, Hans J Skaug, and Guttorm Alendal. A comparison of Monte Carlo sampling methods for metabolic network models. *PLOS One*, 15(7):e0235393, 2020.

20 Andrew Gelman and Donald B. Rubin. Inference from Iterative Simulation Using Multiple Sequences. *Statistical Science*, 7(4):457–472, 1992. Publisher: Institute of Mathematical Statistics. URL: https://www.jstor.org/stable/2246093.

21 Charles J. Geyer. Practical Markov chain Monte Carlo. *Statist. Sci.*, 7(4):473–483, November 1992. doi:10.1214/ss/1177011137.

22 Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(25):723–773, 2012. URL: http://jmlr.org/papers/v13/gretton12a.html.

23 Elena Gryazina and Boris Polyak. Random sampling: Billiard walk algorithm. *European Journal of Operational Research*, 238(2):497–504, 2014. doi:10.1016/j.ejor.2014.03.041.

24 Gaël Guennebaud, Benoît Jacob, et al. *Eigen v3*, 2010. URL: http://eigen.tuxfamily.org.

25 Hulda S Haraldsdóttir, Ben Cousins, Ines Thiele, Ronan MT Fleming, and Santosh Vempala. CHRR: coordinate hit-and-run with rounding for uniform sampling of constraint-based models. *Bioinformatics*, 33(11):1741–1743, 2017.

26 Laurent Heirendt, Sylvain Arreckx, Thomas Pfau, Sebastián N Mendoza, Anne Richelle, Almut Heinken, Hulda S Haraldsdóttir, Jacek Wachowiak, Sarah M Keating, Vanja Vlasov, et al. Creation and analysis of biochemical constraint-based models using the cobra toolbox v. 3.0. *Nature protocols*, 14(3):639–702, 2019.

27 Trey Ideker, Timothy Galitski, and Leroy Hood. A new approach to decoding life: systems biology. *Annual review of genomics and human genetics*, 2(1):343–372, 2001.

28 Fritz John. Extremum Problems with Inequalities as Subsidiary Conditions. In Giorgio Giorgi and Tinne Hoff Kjeldsen, editors, *Traces and Emergence of Nonlinear Programming*, pages 197–215. Springer, Basel, 2014. doi:10.1007/978-3-0348-0439-4_9.

29 Adam Tauman Kalai and Santosh Vempala. Simulated annealing for convex optimization. *Mathematics of Operations Research*, 31(2):253–266, 2006. URL: http://www.jstor.org/stable/25151723.

30 David E Kaufman and Robert L Smith. Direction choice for accelerated convergence in hit-and-run sampling. *Operations Research*, 46(1):84–95, 1998.

31 Zachary A King, Justin Lu, Andreas Dräger, Philip Miller, Stephen Federowicz, Joshua A Lerman, Ali Ebrahim, Bernhard Ø. Palsson, and Nathan E Lewis. Bigg models: A platform for integrating, standardizing and sharing genome-scale models. *Nucleic acids research*, 44(D1):D515–D522, 2016.

32 Edda Klipp, Wolfram Liebermeister, Christoph Wierling, and Axel Kowald. *Systems biology: a textbook*. John Wiley & Sons, 2016.

33 Peter Kohl, Edmund J Crampin, TA Quinn, and Denis Noble. Systems biology: an approach. *Clinical Pharmacology & Therapeutics*, 88(1):25–33, 2010.

34 Aditi Laddha and Santosh Vempala. Convergence of Gibbs Sampling: Coordinate Hit-and-Run Mixes Fast, 2020. arXiv:2009.11338.

35 Yin Tat Lee and Santosh S. Vempala. Convergence rate of riemannian hamiltonian monte carlo and faster polytope volume computation. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2018, page 1115–1121, New York, NY, USA, 2018. Association for Computing Machinery. doi:10.1145/3188745.3188774.

36 László Lovász, Ravi Kannan, and Miklós Simonovits. Random walks and an $O^*(n^5)$ volume algorithm for convex bodies. *Random Structures and Algorithms*, 11:1–50, 1997.

37 László Lovász and Santosh Vempala. Simulated annealing in convex bodies and an $O^*(n^4)$ volume algorithms. *J. Computer & System Sciences*, 72:392–417, 2006.

38 Maximilian Lularevic, Andrew J Racher, Colin Jaques, and Alexandros Kiparissides. Improving the accuracy of flux balance analysis through the implementation of carbon availability constraints for intracellular reactions. *Biotechnology and bioengineering*, 116(9):2339–2352, 2019.

39 Michael MacGillivray, Amy Ko, Emily Gruber, Miranda Sawyer, Eivind Almaas, and Allen Holder. Robust analysis of fluxes in genome-scale metabolic pathways. *Scientific Reports*, 7, December 2017. doi:10.1038/s41598-017-00170-3.

40 Daniel Machado, Sergej Andrejev, Melanie Tramontano, and Kiran Raosaheb Patil. Fast automated reconstruction of genome-scale metabolic models for microbial species and communities. *Nucleic acids research*, 46(15):7542–7553, 2018.

41 Jens Maurer and Steven Watanabe. Boost random number library. Software, 2017. URL: `https://www.boost.org/doc/libs/1_73_0/doc/html/boost_random.html`.

42 Hariharan Narayanan and Piyush Srivastava. On the mixing time of coordinate hit-and-run, 2020. `arXiv:2009.14004`.

43 Denis Noble. *The music of life: biology beyond genes*. Oxford University Press, 2008.

44 Alberto Noronha, Jennifer Modamio, Yohan Jarosz, Elisabeth Guerard, Nicolas Sompairac, German Preciat, Anna Dröfn Daníelsdóttir, Max Krecke, Diane Merten, Hulda S Haraldsdóttir, Almut Heinken, Laurent Heirendt, Stefanía Magnúsdóttir, Dmitry A Ravcheev, Swagatika Sahoo, Piotr Gawron, Lucia Friscioni, Beatriz Garcia, Mabel Prendergast, Alberto Puente, Mariana Rodrigues, Akansha Roy, Mouss Rouquaya, Luca Wiltgen, Alise Žagare, Elisabeth John, Maren Krueger, Inna Kuperstein, Andrei Zinovyev, Reinhard Schneider, Ronan M T Fleming, and Ines Thiele. The Virtual Metabolic Human database: integrating human and gut microbiome metabolism with nutrition and disease. *Nucleic Acids Research*, 47(D1):D614–D624, October 2018. `doi:10.1093/nar/gky992`.

45 Jeffrey D Orth, Ines Thiele, and Bernhard Ø. Palsson. What is flux balance analysis? *Nature biotechnology*, 28(3):245–248, 2010.

46 Bernhard Ø. Palsson. Metabolic systems biology. *FEBS letters*, 583(24):3900–3904, 2009.

47 Bernhard Ø. Palsson. *Systems biology*. Cambridge university press, 2015.

48 Octavio Perez-Garcia, Gavin Lear, and Naresh Singhal. Metabolic network modeling of microbial interactions in natural and engineered environmental systems. *Frontiers in microbiology*, 7:673, 2016.

49 Robert A. Quinn, Jose A. Navas-Molina, Embriette R. Hyde, Se Jin Song, Yoshiki Vázquez-Baeza, Greg Humphrey, James Gaffney, Jeremiah J. Minich, Alexey V. Melnik, Jakob Herschend, Jeff DeReus, Austin Durant, Rachel J. Dutton, Mahdieh Khosroheidari, Clifford Green, Ricardo da Silva, Pieter C. Dorrestein, and Rob Knight. From sample to multi-omics conclusions in under 48 hours. msystems 1: e00038-16. *Crossref, Medline*, 2016.

50 Vivekananda Roy. Convergence Diagnostics for Markov Chain Monte Carlo. *Annual Review of Statistics and Its Application*, 7(1):387–412, 2020. `doi:10.1146/annurev-statistics-031219-041300`.

51 Pedro A. Saa and Lars K. Nielsen. ll-ACHRB: a scalable algorithm for sampling the feasible solution space of metabolic networks. *Bioinform.*, 32(15):2330–2337, 2016. `doi:10.1093/bioinformatics/btw132`.

52 Jan Schellenberger and Bernhard Ø. Palsson. Use of randomized sampling for analysis of metabolic networks. *Journal of biological chemistry*, 284(9):5457–5461, 2009.

53 John R Schramski, Anthony I Dell, John M Grady, Richard M Sibly, and James H Brown. Metabolic theory predicts whole-ecosystem properties. *Proceedings of the National Academy of Sciences*, 112(8):2617–2622, 2015.

54 Siamak S. Shishvan, Andrea Vigliotti, and Vikram S. Deshpande. The homeostatic ensemble for cells. *Biomechanics and Modeling in Mechanobiology*, 17(6):1631–1662, 2018.

55 Robert L. Smith. Efficient Monte Carlo procedures for generating points uniformly distributed over bounded regions. *Operations Research*, 32(6):1296–1308, 1984.

56 Neil Swainston, Kieran Smallbone, Hooman Hefzi, Paul D. Dobson, Judy Brewer, Michael Hanscho, Daniel C. Zielinski, Kok Siong Ang, Natalie J. Gardiner, Jahir M. Gutierrez, Sarantos Kyriakopoulos, Meiyappan Lakshmanan, Shangzhong Li, Joanne K. Liu, Veronica S. Martínez, Camila A. Orellana, Lake-Ee Quek, Alex Thomas, Juergen Zanghellini, Nicole Borth, Dong-Yup Lee, Lars K. Nielsen, Douglas B. Kell, Nathan E. Lewis, and Pedro Mendes. Recon 2.2: from reconstruction to model of human metabolism. *Metabolomics*, 12(7):109, 2016. `doi:10.1007/s11306-016-1051-4`.

57 Ines Thiele and Bernhard Ø. Palsson. A protocol for generating a high-quality genome-scale metabolic reconstruction. *Nature protocols*, 5(1):93, 2010.

A Family of Metrics from the Truncated Smoothing of Reeb Graphs

Erin Wolf Chambers
Department of Computer Science, St. Louis University, MO, USA

Elizabeth Munch
Department of Computational Mathematics,
Science and Engineering and Department of Mathematics,
Michigan State University, East Lansing, MI, USA

Tim Ophelders
Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands

Abstract

In this paper, we introduce an extension of smoothing on Reeb graphs, which we call truncated smoothing; this in turn allows us to define a new family of metrics which generalize the interleaving distance for Reeb graphs. Intuitively, we "chop off" parts near local minima and maxima during the course of smoothing, where the amount cut is controlled by a parameter τ. After formalizing truncation as a functor, we show that when applied after the smoothing functor, this prevents extensive expansion of the range of the function, and yields particularly nice properties (such as maintaining connectivity) when combined with smoothing for $0 \leq \tau \leq 2\varepsilon$, where ε is the smoothing parameter. Then, for the restriction of $\tau \in [0, \varepsilon]$, we have additional structure which we can take advantage of to construct a categorical flow for any choice of slope $m \in [0, 1]$. Using the infrastructure built for a category with a flow, this then gives an interleaving distance for every $m \in [0, 1]$, which is a generalization of the original interleaving distance, which is the case $m = 0$. While the resulting metrics are not stable, we show that any pair of these for $m, m' \in [0, 1)$ are strongly equivalent metrics, which in turn gives stability of each metric up to a multiplicative constant. We conclude by discussing implications of this metric within the broader family of metrics for Reeb graphs.

2012 ACM Subject Classification Mathematics of computing → Algebraic topology; Theory of computation → Computational geometry

Keywords and phrases Reeb graphs, interleaving distance, graphical signatures

Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.22

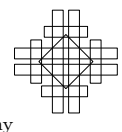
Related Version *Full Version*: https://arxiv.org/abs/2007.07795v3

Funding *Erin Wolf Chambers*: This work was funded in part by the National Science Foundation through grants CCF-1907612 and DBI-1759807.
Elizabeth Munch: This work was funded in part by the National Science Foundation through grants CCF-1907591 and DEB-1904267.
Tim Ophelders: This work was funded in part by the National Science Foundation through grant CCF-1907591.

Acknowledgements The authors wish to thank the anonymous reviewers for their helpful feedback and insights that tightened the bounds of Theorem 5.3.

1 Introduction

The Reeb graph, originally defined in the context of Morse theory [44], represents a portion of the underlying structure of a topological space $\mathbb{X}$ through the lens of a real valued function $h : \mathbb{X} \to \mathbb{R}$; the pair of data $(\mathbb{X}, h)$ is known as an $\mathbb{R}$-space. Specifically, points in the Reeb graph correspond to connected components in the levelsets of the function; as such, the Reeb graph inherits a real valued function from the original input data. For nice enough



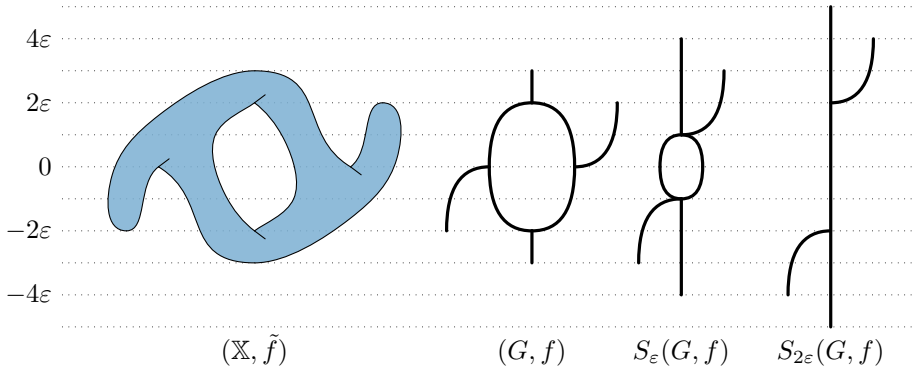


Figure 1 From left to right: an $\mathbb{R}$-space $(\mathbb{X}, \tilde{f})$, its Reeb graph (G, f), smoothings are shown for two parameters, ε and 2ε. Function values are shown by height.

inputs, the resulting object is a finite graph. So, at its core, we focus our study on objects of the form (G, f) where G is a graph and $f : G \to \mathbb{R}$ is a function given on vertices and interpolated linearly on the edges. See Figure 1 for an example.

Reeb graphs have become increasingly useful in a wide range of applications, including settings such as shape comparison [35, 29], denoising [52], shape understanding [25, 34], reconstructing non-linear 1-dimensional structure in data [42, 32, 19, 50], summarizing collections of trajectory data [15], and allowing for informed exploration of otherwise hard-to-visualize high-dimensional data [51, 33]; see [5] for a survey of these and more topics. As a result, there is interest in defining metrics on these objects, to evaluate their quality in the face of noisy input data as well as to allow for more accurate shape comparison and analysis. In this setting, we are focused on metrics that incorporate both the graph and function information: so $d((G, f_1), (G, f_2))$ should be non-zero if $f_1 \neq f_2$ even though they are defined on the same underlying graphs.

Several metrics have arisen recently to do this, taking inspiration from different mathematical backgrounds [23, 4, 3, 16, 27, 28, 1, 47, 2]. In this paper, we focus on the Reeb graph interleaving distance [23]. The basic idea is to work with a notion of smoothing, which returns a parameterized family of Reeb graphs, $S_\varepsilon(G, f)$ for every $\varepsilon \geq 0$, starting with $\varepsilon = 0$ which leaves the input unchanged. This procedure simplifies the loop structures and stretches tails [54]; see Figure 1 for an example. Then the goal is to find an ε-interleaving, which is a pair of families of maps making a particular diagram commute. If $\varepsilon = 0$, this diagram simplifies down to finding an isomorphism between the two Reeb graphs; increasing ε provides more flexibility to find such pairs of maps. Then we have a metric by defining $d_I((G, f), (H_h))$ to be the infimum over the set of ε for which such a diagram exists.

This metric takes root in the interleaving distance defined for persistence modules [18], and is largely inspired by the subsequent category theoretic treatment [14, 12]. This viewpoint comes from encoding the data of a Reeb graph in a constructible set-valued cosheaf [21, 22]. It was later shown that these metrics are special cases of a more general theory of interleaving distances given on a *category with a flow* [24, 48, 20]. This framework encompasses common metrics including ℓ_∞ distance on points or functions, regular Hausdorff distance, and the Gromov-Hausdorff distance [48, 13]. Using this framework, interleaving metrics have been studied in the context of $\mathbb{R}$-spaces [8], multiparameter persistence modules [38], merge trees [39], and formigrams [36, 37], and on more general category theoretic constructions [10, 45]. There are also interesting restrictions to labeled merge trees, where one can pass to a matrix representation and show that the interleaving distance is equivalent to the point-wise ℓ_∞ distance [40, 31, 53, 49].

On the negative side, it has been shown that Reeb graph interleaving is graph isomorphism complete [23, 6], and that many other variants are also NP-hard [6, 7]. All of this means that these metrics, while mathematically interesting, may not lead to feasible algorithms for comparison and analysis. However, a glimmer of hope arises with work investigating fixed parameter tractable algorithms [30, 49]. Despite the issues of computational complexity, notions of similarity for graphs in general, and Reeb graphs in particular, are of pressing interest due to their extensive use in data analysis; in many such settings, we are concerned with questions of quality in the face of noise, and understanding convergence of approximations to a true underlying structure. For example, the interleaving distance has been used in evaluating the quality of the mapper graph [46], which can be proven to be a approximation of the Reeb graph using this metric [41, 11]. Furthermore, there is considerable interest in unifying the interleaving distance with the emerging collection of other Reeb graph metrics.

In this paper, we introduce a truncation operation, which intuitively cuts off portions of the Reeb graph near local extrema with respect to f; this operation is easy to compute for any Reeb graph and tends to result in a simplified Reeb graph. We show that truncation is a functor, and when combined with the smoothing functor, defines a flow on the the category of Reeb graphs. We investigate and prove particularly desirable geometric and topological properties of *truncated smoothing* for certain ranges of the two parameters controlling the functors. We then introduce a new family of metrics for Reeb graphs, called truncated interleaving distances. They are parameterized by $m \in [0, 1]$, and generalize the interleaving distance, with the setting $m = 0$ being the original interleaving distance. We show that the metrics arising from $m \in [0, 1)$ are strongly equivalent. Although the metrics are not stable in the sense of [3], strong equivalence implies that they are at least stable up to a constant.

When combined with preliminary work on geometric implications of smoothing [54], truncated smoothing is interesting in its own right, as it provides a collection of paths for Reeb graph space to be studied in terms of the resulting persistence diagrams. It also is useful when considering algorithms to test planarity for Reeb graphs, or find planar representations of them. The new family of metrics also provide the possibility for new approaches for approximation algorithms for the interleaving distance, as well as new avenues for further unification of the broader family of Reeb graph metrics.

Outline. We give the basic background on Reeb graphs, smoothing, and the Reeb graph interleaving distance in Section 2. Next, we introduce our definition of truncated smoothing in Section 3. In Section 4 we check properties of the truncated smoothing operation. We then take a categorical view of truncated smoothing to develop a family of metrics and investigate their properties in Sections 5 and 6. Finally, the results are discussed in Section 7. Note that many proofs and technical details, as well as full background, several equivalent alternative formulations, and more examples are included in the full version of the paper [17].

2 Background: Reeb graphs, smoothing, and interleaving

Given a topological space $\mathbb{X}$ along with a continuous $\mathbb{R}$-valued function $f\colon \mathbb{X} \to \mathbb{R}$, we call the pair $(\mathbb{X}, f)$ an $\mathbb{R}$-*space*. For two $\mathbb{R}$-spaces $(\mathbb{X}, f)$ and $(\mathbb{X}', g)$, we call a continuous map $\varphi\colon \mathbb{X} \to \mathbb{X}'$ *function-preserving* if $f = g \circ \varphi$, and write $\varphi\colon (\mathbb{X}, f) \to (\mathbb{X}', g)$ in that case.

For an $\mathbb{R}$-space $(\mathbb{X}, f)$, we define an equivalence relation $\sim_f$ on the points of $\mathbb{X}$, such that $x \sim_f x'$ if and only if x and x' lie in the same path-connected component of $f^{-1}(y)$ for some $y \in \mathbb{R}$. For sufficiently nice functions[1], the quotient space $\mathbb{X}/\sim_f$ is a graph,

[1] e.g. a Morse function on a manifold, or a constructible space and function [23], or a space with a levelset-tame function [26].

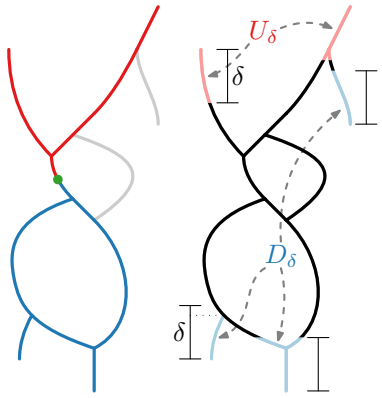


Figure 2 Left: the up-set (red) and down-set (blue) of a point. Although the up-set is a tree, it is not an up-tree as it contains down-forks of the ambient graph. Right: the sets U_δ and D_δ of points with no length δ up-path or down-path, respectively. The leftmost component of D_δ does not contain the down-fork.

called a *Reeb graph*, and we denote the quotient map by $q_f \colon (\mathbb{X}, f) \to (\mathbb{X}/\sim_f, g)$. Since $f(x) = f(x')$ whenever $x \sim_f x'$, we can treat the Reeb graph as an $\mathbb{R}$-space $(X/\sim_f, g)$ by defining $g(q_f(x)) = f(x)$, so that q_f is function-preserving. Most but not all functions in this paper are function preserving. Figure 1 illustrates the construction of a Reeb graph of an $\mathbb{R}$-space.

For the purposes of this work, we will largely divorce the idea of the Reeb graph from the need for a starting space that was used to construct it. Thus for our purposes, a *Reeb graph* is a pair (G, f) where $G = (V_G, E_G)$ is a finite multigraph and $f \colon G \to \mathbb{R}$, referred to as the *height function*, is a continuous map that is linearly interpolated along edges of G, and for which no two neighboring vertices have the same function value. We write $\mathrm{Im}(G, f) := f(G) \subset \mathbb{R}$ for the image of the graph in $\mathbb{R}$. The function can equivalently be stored by defining $f \colon V_G \to \mathbb{R}$ as a function on the vertices, and extending it to the edges implicitly. We treat G as a topological space, so that a point $x \in G$ lies either on a vertex of G, or interior to an edge of G. For succinctness, we also write $x \in (G, f)$ to mean $x \in G$. Since no two adjacent vertices have the same function value, a level set $f^{-1}(y)$ for $y \in \mathbb{R}$ is a finite set of points in G which could be vertices and/or points in the interior edges.

Together, the collection of Reeb graphs (treated as $\mathbb{R}$-spaces) with function-preserving maps as morphisms forms a category, **Reeb**. For the reader without a background in category theory, the basic idea is that that this collection of objects and morphisms satisfy some basic axiomatic structures that make their analysis easier to view as a collection. It also makes available the viewpoint of *functors* between categories, which are essentially structure preserving maps. For now, we will largely hand-wave past the categorical constructions, and defer the technicalities to the full version of the paper.

Define a *path* from x to x' in (G, f) to be a continuous map $\pi \colon [0, 1] \to G$ such that $\pi(0) = x$ and $\pi(1) = x'$. A path is called an *up-path* if it is monotone-increasing with respect to the function, i.e. $f(\pi(t)) \leq f(\pi(t'))$ for $t \leq t'$. Symmetrically, a path is a *down-path* if it is monotone-decreasing. In the case of an up- or down-path π, we call $|f(\pi(0)) - f(\pi(1))|$ the *height* of the path.

In a Reeb graph (G, f), let the *up-paths* of a point x be the set of f-monotone paths that have x as minimum. The *up-set* of a point x is the set of points reachable from x by an up-path, including x itself. Define an *up-fork* to be a vertex x whose up-set contains at least

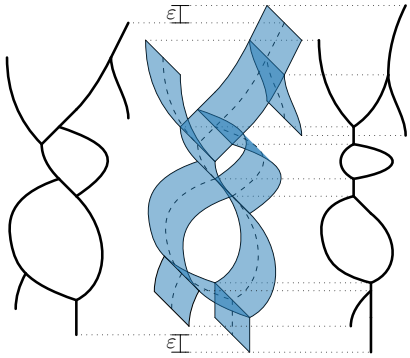


Figure 3 From left to right: a Reeb graph (G, f), its ε-thickening $(G \times [-\varepsilon, \varepsilon], f + \mathrm{Id})$, and the Reeb graph $S_\varepsilon(G, f)$ of the ε-thickening. The product of an edge with an interval is drawn to reflect the function value at a given height.

two edges adjacent to x. We define *down-paths*, *down-sets*, and *down-forks* symmetrically. Call the up-set of a point x an *up-tree* if it contains no down-forks of (G, f), and say that x *roots* an up-tree in such case. The concept of rooting a *down-tree* is defined symmetrically. See Figure 2.

▶ **Definition 2.1.** *Fix a Reeb graph (G, f) and $\varepsilon \geq 0$. Define the* ε-thickening *of G to be the space $G \times [-\varepsilon, \varepsilon]$ with the product topology, and define $(f + \mathrm{Id}) \colon G \times [-\varepsilon, \varepsilon] \to \mathbb{R}$ by $(f + \mathrm{Id})(x, t) = f(x) + t$. We define the* ε-smoothing *$S_\varepsilon(G, f)$ to be the Reeb graph of $(f + \mathrm{Id})$, and denote the corresponding quotient map by $q \colon G \times [-\varepsilon, \varepsilon] \to S_\varepsilon(G, f)$. The composition of q with the the inclusion $G \hookrightarrow G \times [-\varepsilon, \varepsilon]; x \mapsto (x, 0)$ is denoted $\eta = q \circ (\mathrm{Id}, 0)$.*

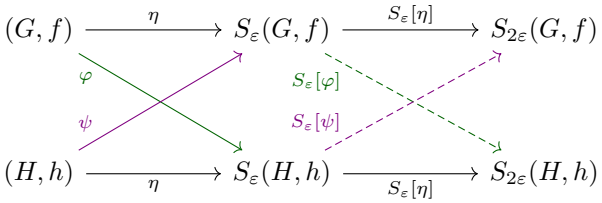
See Figure 3 for an example. In essence, smoothing eliminates small cycles whose height is $\leq 2\varepsilon$, and shrinks all other cycles; it also moves every up-fork and local maximum up and every down-fork and local minimum down. Under the lens of studying the topology of the graph (and in turn the original space), this serves as a functor that can be used to remove noise and simplify topology in a parameterized fashion.

The smoothing construction, S_ε, holds quite a bit more useful structure as not only is it a functor, it is an example of a flow [24]. While we do not provide the full definition here, the specifics are given in the full version of the paper. In particular, this comes from using the additional structure afforded by the function preserving map $\eta \colon (G, f) \to S_\varepsilon(G, f)$. We will reserve the full investigation of η until the full version of the paper, but will use the following property of categories with a flow.

▶ **Theorem 2.2** ([24, Thm. 2.7])**.** *A category with a flow gives rise to an interleaving distance on the objects of the category; specifically, this construction is an extended pseudometric.*

This construction is quite useful since simply by finding some relatively easy to check structure on a category, we immediately get a distance measure on the objects. Depending on the category and flow, this construction encompasses many standard metrics such as the Hausdorff distance; and with a choice of other categories and flows we can construct new metrics. We are particularly interested in the special case of the interleaving distance for Reeb graphs as studied in [23].

▶ **Definition 2.3.** *An* ε-interleaving *with respect to* S_ε *is a pair of maps,* $\varphi\colon (G,f) \to S_\varepsilon(H,h)$ *and* $\psi\colon (H,h) \to S_\varepsilon(G,f)$ *such that the diagram*



commutes. The interleaving distance *is defined to be*

$$d_I((G,f),(H,h)) = \inf_\varepsilon \{\textit{there exists an } \varepsilon\textit{-interleaving of } (G,f) \textit{ and } (H,h)\}.$$

In the construction on this category, d_I is an extended metric since the interleaving distance between Reeb graphs with different numbers of connected components is ∞ as there is no interleaving available for any ε [23]. One particularly useful property we will make use of is understanding how the image of the smoothed Reeb graph, $\mathrm{Im}(S_\varepsilon(G,f)) := f(G) \subseteq \mathbb{R}$, changes under smoothing. Note that if G is connected, $\mathrm{Im}(G,f)$ is connected so it is an interval.

▶ **Proposition 2.4.** *For a connected Reeb graph* (G,f) *with* $\mathrm{Im}(G,f) = [a,b]$,

$$\mathrm{Im}(S_\varepsilon(G,f)) = [a-\varepsilon, b+\varepsilon].$$

Proof. For any $c \in \mathrm{Im}(S_\varepsilon(G,f))$, we show that $c \in [a-\varepsilon, b+\varepsilon]$. There is some $x \in S_\varepsilon(G,f)$ with $f_\varepsilon(x) = c$, where f_ε is the induced function on $S_\varepsilon(G,f)$. Then there is a $(y,t) \in G \times [-\varepsilon,\varepsilon]$ with $f(y) + t = c$. Combining $a \le f(y) \le b$ and $-\varepsilon \le t \le \varepsilon$ gives that $a - \varepsilon \le c \le b + \varepsilon$.

For the other direction, let $c \in [a-\varepsilon, b+\varepsilon]$. There exists some $d \in [a,b]$ with $c - d \in [-\varepsilon,\varepsilon]$. Because $\mathrm{Im}(G,f) = [a,b]$, there exists some $x \in f^{-1}(d)$ and $(x, c-d) \in G \times [-\varepsilon,\varepsilon]$ quotients to some $y \in S_\varepsilon(G,f)$ with $f_\varepsilon(y) = c$, so $\mathrm{Im}(S_\varepsilon(G,f)) = [a-\varepsilon, b+\varepsilon]$. ◀

3 Truncated smoothing

We can now introduce our new, modified smoothing of Reeb graphs. Notice from Proposition 2.4 that as the Reeb graph is smoothed, the image becomes larger. The basic idea of truncated smoothing is to cut off some of those expanding tails in a well-defined way.

Let $U_\tau(G,f)$ be the set of points of G that do not have a length τ up-path, and define $D_\tau(G,f)$ symmetrically for down-paths. Note that for any point $x \in U_\tau(G,f)$, all up-paths from x also lie in $U_\tau(G,f)$; the symmetric property is true for $D_\tau(G,f)$. Both $U_\tau(G,f)$ and $D_\tau(G,f)$ are open subsets of (G,f). See Figure 2 for an example. With this, we can define truncation as follows.

▶ **Definition 3.1.** *The* τ-truncation *of* (G,f)*, is the subgraph of* (G,f) *consisting of the points that have both an up-path and a down-path of height* τ*; specifically*

$$T^\tau(G,f) := (G,f) \setminus (U_\tau(G,f) \cup D_\tau(G,f)).$$

This operation can be seen in the second and third graphs of Figure 4. Notice that $T^0(G,f) = (G,f)$, and that for large enough τ, it is entirely possible to disconnect the graph, or even to be left with an empty graph. Utilizing the truncation operation in conjunction with the Reeb graph smoothing operation is what we call truncated smoothing.

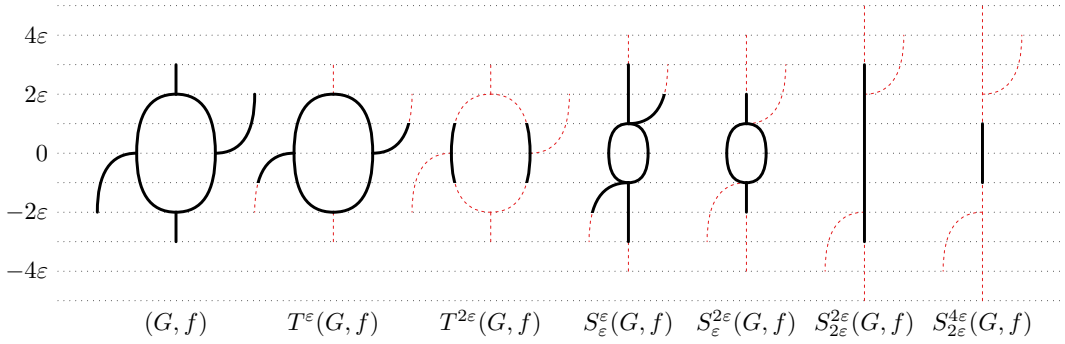


Figure 4 Example of smoothing and truncating for a range of values, on the graph from Figure 1.

▶ **Definition 3.2.** *Let (G, f), $\varepsilon \geq 0$ and $\tau \geq 0$ be given. Then the* truncated smoothing *of (G, f) is defined by $S^\tau_\varepsilon(G, f) = T^\tau S_\varepsilon(G, f)$.*

If $\tau = 0$, $S^0_\varepsilon(G, f) = T^0(S_\varepsilon(G, f)) = S_\varepsilon(G, f)$. So S^0_ε is the same as S_ε, and thus the truncated smoothing can be thought of as a generalization of the smoothing definition.

Consider Figure 4, which shows why we smooth before truncating and more generally, why we will soon want to place restrictions on the relationship between τ and ε. Namely, for this example, we have drawn $T^\varepsilon(G, f)$ and $T^{2\varepsilon}(G, f)$. In the second case in particular, it is clear that truncation has massive detrimental effects on the topology as evidenced by the fact that $T^{2\varepsilon}(G, f)$ has two connected components. However, we can avoid these issues when we smooth first. In the last four examples, smoothing serves to move cycles away from the extrema, so that for a limited amount of truncation, no cycles are broken. We will quantify this "safe" amount of truncation in Section 4. So, while the smoothing parameter still gets rid of the center circle, the truncation only gets rid of expanding tails.

Algorithm. The τ-truncation of a Reeb graph (G, f) can be computed by first storing the length of the longest up-path and down-path of each vertex. This can be done in linear time using a topological sort of the graph based on directing all edges upward. We can for each local maximum store that it has a 0-length up-path, and for the remaining vertices, processes in the order given by the topological sort, storing the length of their up-path based on the stored length of all previously processed neighbors. We store the length of the longest down-path for each vertex symmetrically. Now, we can compute for each edge how much of it remains in the truncation, and subdivide the edges if necessary. Finally, remove all vertices and edges that do not have a sufficiently long up-path or down-path. This procedure takes $O(n + m)$ time on a graph with n vertices at m edges. The truncated smoothing can be computed by first computing the smoothing [23] in $O(m \log(m + n))$ time, giving a total running time of $O(m \log(m + n))$.

4 Properties of truncated smoothing

We can visualize the relationship between τ and ε as drawn in Figure 5. For this figure, we assume we start with a connected Reeb graph (G, f) and study properties of $S^\tau_\varepsilon(G, f)$ which is represented by the point (ε, τ) in the plane. In the remainder of this section, we state the properties of S^τ_ε in different regions of the ε-τ-plane, culminating in the parameter space labeling of Figure 7. We will focus in this section on the case where G is a connected graph, although some results can be modified to incorporate disconnected inputs. These results on disconnected graphs, as well as many of the more technical proofs, are presented in the full version of the paper.

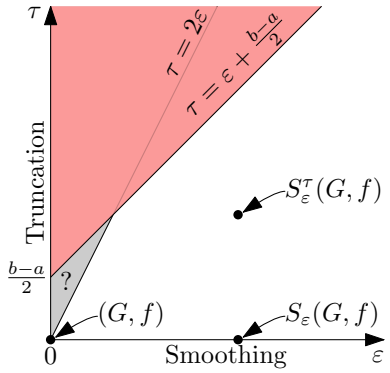


Figure 5 Visualization of Proposition 4.1. Given a connected G where $\mathrm{Im}(G,f) = [a,b] \subset \mathbb{R}$, $S_\varepsilon^\tau(G,f)$ is empty if it is in the red region and non-empty if it is in the white region. Parameters in the grey region can be either empty or not.

4.1 When is $S_\varepsilon^\tau(G,f)$ empty?

We first study the values of ε and τ for which the truncated smoothing is empty. For the purposes of notation, define $\mathrm{Im}(G,f) = f(G) \subset \mathbb{R}$. Consider the following simple example: Let $L_{[a,b]}$ be a Reeb graph consisting of a single edge with image $[a,b] \subseteq \mathbb{R}$, and for an interval $I \subseteq [a,b]$, let $L_I \subseteq L_{[a,b]}$ be the unique subgraph with image I. Then $T^\tau(L_{[a,b]}) = L_{[a+\tau,b-\tau]}$ if $2\tau \leq b-a$, and is the empty Reeb graph for $2\tau > b-a$. On the other hand, $S_\varepsilon(L_{[a,b]})$ is isomorphic to $L_{[a-\varepsilon,b+\varepsilon]}$.

In particular, T^τ and S_ε transform any monotone path with image $[a,b]$ into a monotone path with image $[a+\tau, b-\tau]$, $[a-\varepsilon, b+\varepsilon]$, respectively. In addition, smoothing or truncating the empty Reeb graph again yields the empty Reeb graph. We can build this intuition into the following proposition; details are in the full version of the paper. Note that in the case of a connected graph G, $\mathrm{Im}(G,f)$ is connected and thus is an interval.

▶ **Proposition 4.1.** *Let (G,f) be connected with $\mathrm{Im}(G,f) = [a,b]$.*
- *If $b-a < 2(\tau-\varepsilon)$, then $\mathrm{Im}(S_\varepsilon^\tau(G,f)) = \emptyset$.*
- *If $b-a \geq 2(\tau-\varepsilon)$ and $\tau \leq 2\varepsilon$, then $\mathrm{Im}(S_\varepsilon^\tau(G,f)) = [a-(\varepsilon-\tau), b+(\varepsilon-\tau)]$.*

Sketch proof. We first show that $b-a < 2(\tau-\varepsilon)$ implies the image is empty. We show in the full version of the paper that for a connected graph (H,h) with image $[a',b']$ and $b'-a' < 2\tau$, $T^\tau(H,h)$ is empty. By Proposition 2.4, $\mathrm{Im}(S_\varepsilon(G,f)) = [a-\varepsilon, b+\varepsilon]$. Then setting $S_\varepsilon(G,f) = (H,h)$, we have for $b-a < 2(\tau-\varepsilon)$, that $(b+\varepsilon)-(a-\varepsilon) \leq 2\tau$, so $\mathrm{Im}(S_\varepsilon^\tau(G,f)) = \mathrm{Im}(T^\tau(S_\varepsilon(G,f))) = \emptyset$.

Now, we can assume $b-a \geq 2(\tau-\varepsilon)$. One direction of containment is easy since by Proposition 2.4, $\mathrm{Im}(S_\varepsilon^\tau(G,f)) = \mathrm{Im}(T^\tau(S_\varepsilon(G,f))) \subseteq [a-(\varepsilon-\tau), b+(\varepsilon-\tau)]$. Thus, it remains to show that $[a-(\varepsilon-\tau), b+(\varepsilon-\tau)] \subseteq \mathrm{Im}(S_\varepsilon^\tau(G,f))$. The basic idea is to take two points $s,t \in S_\varepsilon(G,f)$ with $f(s) = a-\varepsilon$ and $f(t) = b+\varepsilon$, and show that they are connected by a path π in $S_\varepsilon(G,f)$ for which the only portions that get truncated are the endpoints. This is simple if π is itself a monotone path; otherwise we use the fact that G has already been smoothed and that we do not truncate too much ($\tau \leq 2\varepsilon$) to show that the parts of the path which are not monotone still have long enough up- and down-paths to not be removed. ◀

This proposition gives us that $S_\varepsilon^\tau(G,f)$ is an empty graph if (ε,τ) is interior to the red region of Figure 5, and is empty in the white region. We cannot expand this proposition to the grey region of Figure 5 as there are examples for which $S_\varepsilon^\tau(G,f)$ can be either empty or

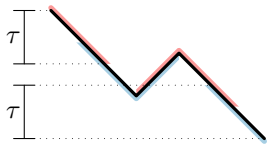


Figure 6 A Reeb graph (G, f) for which $T^\tau(G, f) = S_0^\tau(G, f)$ is empty. This choice of τ is such that $\mathrm{Im}(G, f)$ has diameter greater than 2τ, thus $S_0^\tau(G, f)$ is in the grey region of Figure 5.

not. For instance, in the example of Figure 6, $|\mathrm{Im}(G, f)| \geq 2(\tau - \varepsilon)$, but each position in the graph is either missing a long enough up- or down-path, and hence the truncated graph is empty. On the other hand, for the graph with a single edge $L_{[a,b]}$, any truncation $\tau < \frac{b-a}{2}$ is non empty.

4.2 When does $S_\varepsilon^\tau(G, f)$ maintain connectivity?

Our next goal is to understand when truncation preserves the connectivity of the input. As seen in Figure 4, clearly just truncating the graph can disconnect an originally connected graph. However, what is interesting is that smoothing first and not truncating too much relative to the smoothing will maintain the connectivity; this will be made precise in Proposition 4.6. For this, we introduce two properties, *t-tailed* and *s-safe*, and study how they are affected by smoothing and truncation.

▶ **Definition 4.2.** *A Reeb graph is* t-tailed *if it has a height t up-path at every down-fork and a length t down-path at every up-fork. A Reeb graph is* weakly s-safe *if each component has a point with both an up-path and a down-path of height at least s. A Reeb graph is* s-safe *if it is both s-tailed and weakly s-safe.*

Note that every non-empty Reeb graph is 0-safe. For example, the graph drawn in Figure 2 is not δ-tailed because the bottommost up-fork has no down-path of height δ; in addition, the topmost down-fork has no up-path of height δ.

We next have two results, proved in the full version of the paper, which show how the $\bullet$-tailed and $\bullet$-safe properties are maintained under smoothing and truncating, albeit with modified parameters.

▶ **Proposition 4.3.** *If (G, f) is t-tailed, then $S_\varepsilon(G, f)$ is $(t + 2\varepsilon)$-tailed. If (G, f) is s-safe, then $S_\varepsilon(G, f)$ is $(s + \varepsilon)$-safe. In particular, $S_\varepsilon(G, f)$ is always 2ε-tailed and ε-safe.*

▶ **Lemma 4.4.** *Fix $0 \leq \tau \leq \varepsilon$. If (G, f) is ε-tailed or safe, then $T^\tau(G, f)$ is $(\varepsilon - \tau)$-tailed or safe, respectively.*

Combining Proposition 4.3 and Lemma 4.4, we can see that outside the pink and grey regions of Figure 7, we know that $S_\varepsilon^\tau(G, f)$ is $(t + 2\varepsilon - \tau)$-tailed and $(s + \varepsilon - \tau)$-safe.

▶ **Proposition 4.5.** *Fix $0 \leq \varepsilon$ and $0 \leq \tau$, and assume (G, f) is t-tailed and s-safe. If $\tau \leq t + 2\varepsilon$ and $\tau \leq \varepsilon + \|\mathrm{Im}(G, f)\|/2$, then $S_\varepsilon^\tau(G, f)$ is $(t + 2\varepsilon - \tau)$-tailed and $(s + \varepsilon - \tau)$-safe.*

Proof. Because (G, f) is t-tailed, $S_\varepsilon(G, f)$ is $(t + 2\varepsilon)$-tailed by the first statement of Proposition 4.3. Since $\tau \leq t + 2\varepsilon$, $S_\varepsilon^\tau(G, f) = T^\tau S_\varepsilon(G, f)$ is $(t + 2\varepsilon - \tau)$-tailed by Lemma 4.4. Similarly, since (G, f) is s-safe, $S_\varepsilon(G, f)$ is $(t + \varepsilon)$-safe by the second statement of Proposition 4.3. Then since $\tau \leq t + 2\varepsilon$, $S_\varepsilon^\tau(G, f) = T^\tau S_\varepsilon(G, f)$ is $(s + \varepsilon - \tau)$-safe by Lemma 4.4. ◀

This brings us to our conclusion of parameters for which the connectivity is maintained, with full details provided in the full version of the paper.

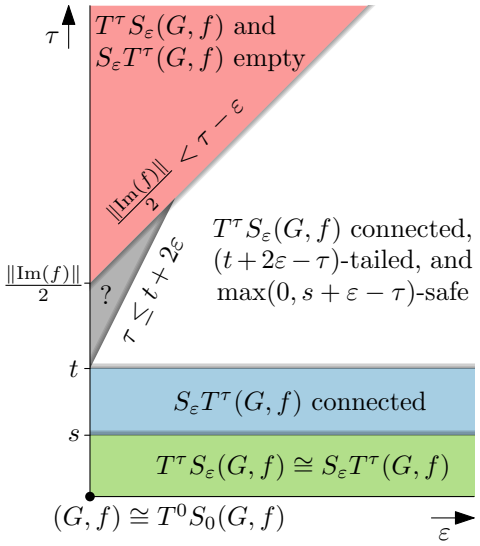


Figure 7 For connected, t-tailed, and s-safe (G,f), properties of $S^\tau_\varepsilon(G,f) = T^\tau S_\varepsilon(G,f)$ and $S_\varepsilon T^\tau(G,f)$, parameterized by τ and ε.

▶ **Proposition 4.6.** *If (G,f) is connected and $\tau \in [0, 2\varepsilon]$, then $S^\tau_\varepsilon(G,f)$ is also connected.*

Sketch proof. We show in the full version of the paper that for a connected, t-tailed graph, $T^t(G,f)$ is connected by ensuring disjointness of the portion of the graph G removed because it is lacking an up-path, and that which is removed because it is lacking a down-path. The result is then a corollary of Proposition 4.3. ◀

4.3 When do S^ε and T^τ commute?

We finally investigate the commutativity of smoothing and truncating. The example of Figure 4 shows why we must be careful with order of operations since $T^\tau S_\varepsilon(G,f)$ is not necessarily the same as $S_\varepsilon T^\tau(G,f)$. Specifically, $S^{2\varepsilon}_{2\varepsilon}(G,f) = T^{2\varepsilon} S_{2\varepsilon}(G,f)$ has one connected component, but any smoothing of $T^{2\varepsilon}(G,f)$ has two connected components. However, the next two results imply that this issue does not arise if we smooth sufficiently before truncating.

▶ **Proposition 4.7.** *If (G,f) is τ-safe, then $S_\varepsilon T^\tau(G,f) \cong T^\tau S_\varepsilon(G,f)$.*

The proof is provided in the full version of the paper. Combining the proposition with Lemma 4.4 and Proposition 4.3 gives the surprising result that the functors T and S do commute in the green region of Figure 7. We can next use this result to show that for certain choices of ε and τ, we can additively combine the parameters for truncated smoothing.

▶ **Theorem 4.8.** *If (1) (G,f) is empty or (2) $\tau_1 \le 2\varepsilon_1$ and (G,f) is weakly $(\tau_1 - \varepsilon_1)$-safe, then $S^{\tau_2}_{\varepsilon_2} S^{\tau_1}_{\varepsilon_1}(G,f) \cong S^{\tau_1+\tau_2}_{\varepsilon_1+\varepsilon_2}(G,f)$.*

Proof. Both smoothing and truncating the empty Reeb graph yields the empty Reeb graph. So we are done if (G,f) is the empty Reeb graph, and we obtain not only an isomorphism but an equality. Now suppose that (G,f) is not empty. Then $S_{\varepsilon_1}(G,f)$ is $2\varepsilon_1$-tailed and weakly $(\tau_1 - \varepsilon_1 + \varepsilon_1)$-safe, and by definition $\min(2\varepsilon_1, \tau_1) \ge \tau_1$-safe. Therefore $S_{\varepsilon_2} T^{\tau_1} S_{\varepsilon_1}(G,f) \cong T^{\tau_1} S_{\varepsilon_2} S_{\varepsilon_1}(G,f)$, and hence using Proposition 4.7,

$$S^{\tau_2}_{\varepsilon_2} S^{\tau_1}_{\varepsilon_1}(G,f) = T^{\tau_2} S_{\varepsilon_2} T^{\tau_1} S_{\varepsilon_1}(G,f) \cong T^{\tau_2} T^{\tau_1} S_{\varepsilon_2} S_{\varepsilon_1}(G,f) \cong S^{\tau_1+\tau_2}_{\varepsilon_1+\varepsilon_2}(G,f). \qquad \blacktriangleleft$$

In particular, the assumptions of the theorem are satisfied if $\tau_1 \le \varepsilon_1$ since every non-empty graph is 0-safe.

5 Truncated interleaving distance

In this section, we survey the results related to defining the family of truncated interleaving distances, proving that certain linear subspaces of our two parameter functor space (shown in Figure 7) form a categorical flow. Since any category with a flow gives an interleaving distance, we then use truncated smoothing to build a new family of metrics for Reeb graphs.

The whole idea behind building a category with a flow is that the flow itself must be functorial, which means we must have knowledge of how it acts both on objects and morphisms. So far, the results discussed in Section 4 only correspond to the object information. In the full version of the paper, we will describe how to explicitly build the morphisms $S^{\tau}_{\varepsilon}(G,f) \to S^{\tau'}_{\varepsilon'}(G,f)$ (i.e., function preserving maps). However, these morphisms are only available for certain choices of parameters. Restricting our view only to (ε,τ) pairs for which these morphisms exist gives us that for any choice of $m \in [0,1]$ we can set $\tau = m\varepsilon$ to get a flow.

▶ **Theorem 5.1.** *For any $m \in [0,1]$, the map $S^m\colon ([0,\infty),\leq) \to \mathbf{End}(\mathbf{Reeb}); \varepsilon \mapsto S^{m\varepsilon}_{\varepsilon}$ is a functor and defines a categorical flow on* **Reeb**.

Essentially, this m can be thought of as defining the slope of a line based at the origin in the parameter space of Figure 7, and thus using Theorem 2.2, we have an interleaving distance for any line with slope less than 1.

▶ **Corollary 5.2.** *For any $m \in [0,1]$, S^m gives rise to an interleaving-type distance*

$$d_I^m((G,f),(H,h)) := \inf\{\varepsilon \geq 0 \mid \text{there exists a } \varepsilon\text{-interleaving with respect to } S^m\}.$$

Specifically, d_I^m is an extended pseudo-metric.

In the next theorem, we show that with the exception of $m = 1$, all the metrics created are closely related in the following sense. Two metrics d_A and d_B are said to be *strongly equivalent* if there are positive constants α_1 and α_2 such that $\alpha_1 d_A \leq d_B \leq \alpha_2 d_A$. In the following theorem, we show that d_I^m and $d_I^{m'}$ are strongly equivalent if (m,m') is contained in the white region of Figure 8.

▶ **Theorem 5.3.** *For any pair $m, m' \in [0,1)$ with $0 \leq m' - m < 1 - m'$ the metrics d_I^m and $d_I^{m'}$ are strongly equivalent. Specifically, given Reeb graphs (G,f) and (H,h),*

$$d_I^m((G,f),(H,h)) \leq d_I^{m'}((G,f),(H,h)) \leq \frac{1-m}{1-m'} d_I^m((G,f),(H,h))$$

The proof of this theorem is contained in the full version of the paper. Of course, as long as we are willing to loosen the bounds, this result extends to any pair of $m, m' \in [0,1)$.

▶ **Corollary 5.4.** *For all pairs $0 \leq M \leq M' < 1$, there exist positive constants C_1 and C_2 dependent on M and M' such that*

$$C_1 d_I^M((G,f),(H,h)) \leq d_I^{M'}((G,f),(H,h)) \leq C_2 d_I^M((G,f),(H,h)),$$

and thus d_I^M and $d_I^{M'}$ are strongly equivalent metrics.

Proof. Consider M, M' given with $M \leq M'$. If $M' \leq \frac{1+M}{2}$, then Theorem 5.3 applies directly. Otherwise, we assume that $M' \geq \frac{1+M}{2}$. Then d_I^M is equivalent to d_I^{α} for any α in the interval $(M, \frac{1+M}{2})$ and $d_I^{M'}$ is equivalent to d_I^{β} for any β in the interval $(2M'-1, M')$. Then there is a zigzag like the example in Figure 8 between α and β which remains in the white region and for which each adjacent pair are strongly equivalent metrics. Equivalence of metrics is transitive, so this implies d_I^M and $d_I^{M'}$ are equivalent. ◀

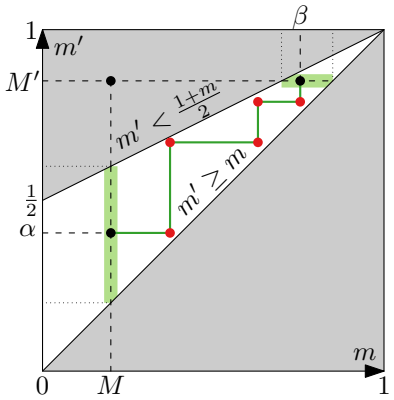


Figure 8 Parameter space for comparing metrics d_I^m and $d_I^{m'}$. The white region is allowable pairs for Theorem 5.3. The vertices of the zigzag (shown as red points) give pairs of strongly equivalent metrics which, when combined, show that M and M' are strongly equivalent in Corollary 5.4.

In particular, this corollary gives that the original Reeb graph interleaving distance (where $m = 0$) is strongly equivalent to d_I^m for all $m \in [0, 1)$. We note that there are many possible zigzag paths which can be used to obtain this bound, but further exploration is needed to determine which, if any, provide optimal constants.

6 Properties of the metrics

As noted, d_I^m is an extended pseudometric, which means it is possible for $d_I^m((G, f), (H, h))$ to be infinite. However, it turns out this is not the case for broad classes of graphs. In fact, in order to take infinite value, there must be no ε-interleaving with respect to S^m between the two Reeb graphs. That being said, there are very specific instances where this metric takes on infinite value.

The easiest case to handle is when $m \in [0, 1)$, since we can use the characterization given in [23] in conjunction with the equivalence of metrics Corollary 5.4.

▶ **Proposition 6.1.** *Let $m \in [0, 1)$. Then $d_I^m((G, f), (H, h)) < \infty$ iff G and H have the same number of path-connected components.*

Proof. Note that $d_I^0 = d_I$. By [23, Prop. 4.5], $d_I((G, f), (H, h))$ is finite if and only if G and H have the same number of path connected components. This combined with Corollary 5.4 gives the proposition. ◀

The characterization of when d_I^m is infinite for $m = 1$ is more complicated. Consider a connected graph (G, f) with $\mathrm{Im}(G, f) = [a, b]$. When $m = 1$, we are interested in understanding the behavior of $S_\varepsilon^\varepsilon(G, f)$. By Proposition 4.1, we see that $b - a \geq 2(\tau - \varepsilon) = 0$, so $S_\varepsilon^\varepsilon(G, f) = [a, b]$. That is to say that the image of (G, f) is unchanged by $S_\varepsilon^\varepsilon$. Now, if we wanted to determine the interleaving distance d_I^1 for a given (G, f) and (H, h), one requirement is always that we must smooth the given graphs enough for there to be a morphism $(G, f) \to S_\varepsilon^\varepsilon(H, h)$. However, because $S_\varepsilon^\varepsilon$ does not change the image, the function preserving requirement of morphisms mean that if the graphs did not start with the same image no choice of ε will make this possible. With this example in mind, we can characterize when d_I^m takes on infinite values for $m = 1$.

▶ **Proposition 6.2.** *Let $m = 1$ and assume G and H are connected. Then*

$$d_I^m((G,f),(H,h)) < \infty \text{ if and only if } \mathrm{Im}(G,f) = \mathrm{Im}(H,h).$$

Further, if $\mathrm{Im}(G,f) = \mathrm{Im}(H,h)$, then $d_I^m((G,f),(H,h)) \leq |\mathrm{Im}(G,f)|$.

Proof. Note that by Proposition 4.1, for any connected G' with $\mathrm{Im}(G',f') = [a,b]$, we have $\mathrm{Im}S_\varepsilon^\varepsilon(G',f') = [a,b]$. So the truncated smoothing maintains the image for every connected component, and thus for the union of the connected components. Thus, we have $\mathrm{Im}(G,f) = \mathrm{Im}(S_\varepsilon^\varepsilon(G,f))$ and $\mathrm{Im}(H,h) = \mathrm{Im}(S_\varepsilon^\varepsilon(H,h))$ for any choice of ε.

Assume we have an $S_\varepsilon^\varepsilon$ interleaving $\varphi\colon (G,f) \to S_\varepsilon^\varepsilon(H,h)$ and $\psi\colon (H,h) \to S_\varepsilon^\varepsilon(G,f)$. Because φ and ψ are function preserving, $\varphi(G) = \mathrm{Im}(G,f) \subseteq \mathrm{Im}(S_\varepsilon^\varepsilon(H,h))$ and $\psi(H) = \mathrm{Im}(H,h) \subseteq \mathrm{Im}(S_\varepsilon^\varepsilon(G,f))$. But since $S_\varepsilon^\varepsilon$ leaves the images unchanged, this implies that $\mathrm{Im}(G,f) = \mathrm{Im}(H,h)$.

Now assume $\mathrm{Im}(G,f) = \mathrm{Im}(H,h)$. Let $\varepsilon = |\mathrm{Im}(G,f)|$ and consider the thickening $G \times [-\varepsilon,\varepsilon]$ and a value $a \in \mathrm{Im}(G,f)$. We claim that $(f+\mathrm{Id})^{-1}(a) \subseteq G \times [-\varepsilon,\varepsilon]$ is exactly $A = \{(x, a-f(x)) \mid x \in G\}$ and in particular, that it is homeomorphic to G. Indeed, for any $x \in G$, $a - f(x) \in [-\varepsilon,\varepsilon]$ and the point $y = (x, a-f(x))$ has image $(f+\mathrm{Id})(y) = a$ so $A \subseteq (f+\mathrm{Id})^{-1}(a)$. Moreover, for any $(x,t) \in (f+\mathrm{Id})^{-1}(a)$, $f(x)+t = a$ so $t = a - f(x)$, thus $(f+\mathrm{Id})^{-1}(a) \subseteq A$.

So, since G is connected and f is continuous, $(f+\mathrm{Id})^{-1}(a) \cong G$ is a single connected component for any $a \in \mathrm{Im}(G,f)$, and the same is true for (H,h). Because the $S_\varepsilon^\varepsilon$ smoothing maintains the image, this implies $S_\varepsilon^\varepsilon(G,f) = S_\varepsilon^\varepsilon(H,f)$ is a single line segment with the same image. We obtain an interleaving by simply sending every point in (G,f) to the unique point at the same height in $S_\varepsilon(H,h)$ and vice versa, so the d_I^m distance is finite. ◀

We next investigate stability, for this collection of metrics.

▶ **Definition 6.3.** *Let $(\mathbb{X},f)$ and $(\mathbb{X},g)$ be $\mathbb{R}$-spaces with the same total space $\mathbb{X}$, and let $R(\mathbb{X},f)$ and $R(\mathbb{Y},g)$ be the respective Reeb graphs. A metric d is said to be stable if*

$$d(R(\mathbb{X},f), R(\mathbb{X},g)) \leq \|f-g\|_\infty.$$

The original Reeb interleaving distance, $m = 0$, is stable [23, Thm 4.4]. Unfortunately, d_I^m is not stable in the strictest sense; to see why, consider the following simple example. Consider two simple line segments for graphs, for example, (L,f_1) and (L,f_2) where $\mathrm{Im}(L,f_1) = [-a,a]$ and $\mathrm{Im}(L,f_2) = [-b,b]$ for $a < b$. Then $\|f_1 - f_2\|_\infty = b - a$. However, the interleaving distance requires that we smooth at least until $[-b,b] = \mathrm{Im}(L,f_2) \subseteq \mathrm{Im}(S_\varepsilon(L,f_1))$. But by Proposition 4.1, $\mathrm{Im}(S_\varepsilon(L,f_1)) = [a-(\varepsilon - m\varepsilon), a+(\varepsilon - m\varepsilon)]$. Thus $d_I^m(f_1,f_2) \geq \frac{b-a}{1-m} \geq b-a$, and is strictly greater if $m \neq 0$. This means that $b - a = \|f_1 - f_2\|_\infty < d_I^m((L,f_1),(L,f_2))$, and thus d_I^m is not stable.

We can regain at least partial control of the distance, however, as d_I^m is still Lipschitz when given a fixed choice of m.

▶ **Proposition 6.4.** *Let $m \in [0,1)$. Assume $(\mathbb{X},g_1)$ and $(\mathbb{X},g_2)$ are given for a connected space $\mathbb{X}$ and denote the associated Reeb graphs by (G,f) and (H,h) respectively. Then there is a positive constant C dependent on m for which*

$$d_I^m((G,f),(H,h)) \leq C\|g_1 - g_2\|_\infty.$$

Proof. By Corollary 5.4, d_I^0 and d_I^m are strongly equivalent metrics, so there is a positive constant C for which $d_I^m \leq C d_I^0$. Then because the Reeb graph interleaving distance d_I^0 is stable, we have

$$d_I^m((G,f),(H,h)) \leq C d_I^0((G,f),(H,h)) \leq C\|g_1 - g_2\|. \quad ◀$$

Because of the dependence on Corollary 5.4 where the optimal choice of zigzag to find the constant C is unclear, we do not give an explicit formulation here. We conclude by connecting our extended pseudometric to two other metrics for Reeb graphs, the functional distortion distance [2] and the bottleneck distance [43]. The proof is a straightforward implication of inequalities, so due to space constraints we simply state these results without formally defining either. The interested reader can find further details on the metrics in [2] and [43].

▶ **Proposition 6.5.** *The truncated interleaving distance is strongly equivalent to the functional distortion distance. Further, defining d_B as the bottleneck distance of the level set persistent homology, we have the inequality $d_B \leq 5d_I^m$.*

Proof. The interleaving distance, d_I^0, is strongly equivalent to the functional distortion distance by [4, Thm 16]. So by Corollary 5.4 and transitivity of strong equivalence, they are each strongly equivalent to d_I^m for any $m \in [0, 1)$. To obtain the inequality, we use the bound on the bottleneck distance of level set persistent homology by the Reeb graph interleaving distance in [9, Thm. 4.13]. ◀

7 Conclusion and discussion

Our primary aim has been to introduce the concept of truncated smoothing and establish properties and connections of this operation. We have several reasons for considering this as a similarity measure on Reeb graphs. First, it has potential for providing bounds for the stable interleaving distance via the equivalence of metrics. Second, we came to this definition while investigating drawings of Reeb graphs and when planarity is achievable (that, is whether a Reeb graph has a planar drawing which respects the function in the y-coordinate). In a subsequent paper, we will show that while traditional smoothing does not maintain planarity, the truncated smoothing does for $\varepsilon \leq \tau \leq 2\varepsilon$.

We suspect additional potential applications of truncated smoothing in comparing geometric or planar graphs, since it simplifies the graph's topology (via smoothing) without suffering from extensive expansion of the co-domain or destruction of desirable combinatorial properties like level planarity. Truncated smoothing also allows for interesting manipulation of the extended persistence diagram of the Reeb graph and computation of morphs between Reeb graphs; again, we defer details to future work, as a full classification of that manipulation is necessary.

We suspect that the loss of stability discussed in Section 6 is not as dire as it seems. If nothing else, Proposition 6.4 gives a Lipschitz constant in advance dependent only on m, so it is possible to upper bound the difference using these new interleaving distances.

While we are able to connect our collection of metrics to several Reeb metrics (Proposition 6.5), we have not investigated further connections to other metrics as of yet. One particularly interesting future direction is to determine whether this collection of metrics provides results related to strong equivalence between the interleaving distance and the universal distance of [3]. Perhaps this broader collection of metrics will help to provide stronger bounds between the various metrics on Reeb graphs, since strong equivalence with one is strong equivalence with all.

References

1 Ulrich Bauer, Barbara Di Fabio, and Claudia Landi. An edit distance for Reeb graphs. In A. Ferreira, A. Giachetti, and D. Giorgi, editors, *Eurographics Workshop on 3D Object Retrieval*. The Eurographics Association, 2016. doi:10.2312/3dor.20161084.

2 Ulrich Bauer, Xiaoyin Ge, and Yusu Wang. Measuring distance between Reeb graphs. In *Proceedings of the Thirtieth Annual Symposium on Computational Geometry - SoCG '14, Kyoto, Japan*, 2014.

3 Ulrich Bauer, Claudia Landi, and Facundo Mémoli. The Reeb Graph Edit Distance Is Universal. In Sergio Cabello and Danny Z. Chen, editors, *36th International Symposium on Computational Geometry (SoCG 2020)*, volume 164 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 15:1–15:16, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2020.15.

4 Ulrich Bauer, Elizabeth Munch, and Yusu Wang. Strong equivalence of the interleaving and functional distortion metrics for Reeb graphs. In Lars Arge and János Pach, editors, *31st International Symposium on Computational Geometry (SoCG 2015)*, volume 34 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 461–475, Dagstuhl, Germany, 2015. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.SOCG.2015.461.

5 S. Biasotti, D. Giorgi, M. Spagnuolo, and B. Falcidieno. Reeb graphs for shape analysis and applications. *Theoretical Computer Science: Computational Algebraic Geometry and Applications*, 392(13):5–22, 2008. doi:10.1016/j.tcs.2007.10.018.

6 Håvard Bakke Bjerkevik and Magnus Bakke Botnan. Computational Complexity of the Interleaving Distance. In Bettina Speckmann and Csaba D. Tóth, editors, *34th International Symposium on Computational Geometry (SoCG 2018)*, volume 99 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 13:1–13:15, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.SoCG.2018.13.

7 Håvard Bakke Bjerkevik, Magnus Bakke Botnan, and Michael Kerber. Computing the interleaving distance is NP-hard. *Foundations of Computational Mathematics*, November 2019. doi:10.1007/s10208-019-09442-y.

8 Andrew J. Blumberg and Michael Lesnick. Universality of the homotopy interleaving distance. *CoRR*, 2017. arXiv:1705.01690v1.

9 Magnus Botnan and Michael Lesnick. Algebraic stability of zigzag persistence modules. *Algebraic & Geometric Topology*, 18(6):3133–3204, October 2018. doi:10.2140/agt.2018.18.3133.

10 Magnus Bakke Botnan, Justin Curry, and Elizabeth Munch. A relative theory of interleavings. *CoRR*, 2020. arXiv:2004.14286v1.

11 Adam Brown, Omer Bobrowski, Elizabeth Munch, and Bei Wang. Probabilistic convergence and stability of random mapper graphs. *Journal of Applied and Computational Topology*, December 2020. doi:10.1007/s41468-020-00063-x.

12 Peter Bubenik, Vin de Silva, and Jonathan Scott. Metrics for generalized persistence modules. *Foundations of Computational Mathematics*, 2014.

13 Peter Bubenik, Vin de Silva, and Jonathan Scott. Interleaving and Gromov-Hausdorff distance. *CoRR*, 2017. arXiv:1707.06288v3.

14 Peter Bubenik and Jonathan A. Scott. Categorification of persistent homology. *Discrete & Computational Geometry*, 51(3):600–627, 2014. doi:10.1007/s00454-014-9573-x.

15 Kevin Buchin, Maike Buchin, Marc van Kreveld, Bettina Speckmann, and Frank Staals. Trajectory grouping structure. In Frank Dehne, Roberto Solis-Oba, and Jörg-Rüdiger Sack, editors, *Algorithms and Data Structures*, volume 8037 of *Lecture Notes in Computer Science*, pages 219–230. Springer Berlin Heidelberg, 2013. doi:10.1007/978-3-642-40104-6_19.

16 Mathieu Carrière and Steve Oudot. Local equivalence and intrinsic metrics between Reeb graphs. In Boris Aronov and Matthew J. Katz, editors, *33rd International Symposium on Computational Geometry (SoCG 2017)*, volume 77 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 25:1–25:15, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.SoCG.2017.25.

17 Erin Wolf Chambers, Elizabeth Munch, and Tim Ophelders. A family of metrics from the truncated smoothing of Reeb graphs. *CoRR*, 2020. arXiv:2007.07795v3.

18 Frédéric Chazal, David Cohen-Steiner, Marc Glisse, Leonidas J. Guibas, and Steve Y. Oudot. Proximity of persistence modules and their diagrams. In *Proceedings of the 25th annual symposium on Computational geometry*, SCG '09, pages 237–246, New York, NY, USA, 2009. ACM. doi:10.1145/1542362.1542407.

19 Frédéric Chazal and Jian Sun. Gromov-Hausdorff approximation of filament structure using Reeb-type graph. In *Proceedings of the Thirtieth Annual Symposium on Computational*

Geometry, SOCG'14, pages 491:491–491:500, New York, NY, USA, 2014. ACM. doi:10.1145/2582112.2582129.

20 Joshua Cruz. Metric limits in categories with a flow. *CoRR*, 2019. arXiv:1901.04828v1.

21 Justin Curry. *Sheaves, Cosheaves and Applications*. PhD thesis, University of Pennsylvania, 2014. arXiv:1303.3255.

22 Justin Curry and Amit Patel. Classification of constructible cosheaves. *CoRR*, 2016. arXiv:1603.01587v5.

23 Vin de Silva, Elizabeth Munch, and Amit Patel. Categorified Reeb graphs. *Discrete & Computational Geometry*, pages 1–53, 2016. doi:10.1007/s00454-016-9763-9.

24 Vin de Silva, Elizabeth Munch, and Anastasios Stefanou. Theory of interleavings on categories with a flow. *Theory and Applications of Categories*, 33(21):583–607, 2018. URL: http://www.tac.mta.ca/tac/volumes/33/21/33-21.pdf.

25 Tamal K. Dey, Fengtao Fan, and Yusu Wang. An efficient computation of handle and tunnel loops via Reeb graphs. *ACM Trans. Graph.*, 32(4):32:1–32:10, 2013. doi:10.1145/2461912.2462017.

26 Tamal K. Dey and Yusu Wang. Reeb graphs: Approximation and persistence. *Discrete & Computational Geometry*, 49(1):46–73, 2013. doi:10.1007/s00454-012-9463-z.

27 B. Di Fabio and C. Landi. Reeb graphs of curves are stable under function perturbations. *Mathematical Methods in the Applied Sciences*, 35(12):1456–1471, 2012. doi:10.1002/mma.2533.

28 Barbara Di Fabio and Claudia Landi. The edit distance for Reeb graphs of surfaces. *Discrete & Computational Geometry*, 55(2):423–461, January 2016. doi:10.1007/s00454-016-9758-6.

29 Francisco Escolano, Edwin R. Hancock, and Silvia Biasotti. Complexity fusion for indexing Reeb digraphs. In Richard Wilson, Edwin Hancock, Adrian Bors, and William Smith, editors, *Computer Analysis of Images and Patterns*, volume 8047 of *Lecture Notes in Computer Science*, pages 120–127. Springer Berlin Heidelberg, 2013. doi:10.1007/978-3-642-40261-6_14.

30 Elena Farahbakhsh Touli and Yusu Wang. FPT-algorithms for computing Gromov-Hausdorff and interleaving distances between trees. In *27th Annual European Symposium on Algorithms, ESA 2019, September 9-11, 2019, Munich/Garching, Germany*, volume 144 of *LIPIcs*, pages 83:1–83:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPICS.ESA.2019.83.

31 Ellen Gasparovic, Elizabeth Munch, Steve Oudot, Katharine Turner, Bei Wang, and Yusu Wang. Intrinsic interleaving distance for merge trees. *CoRR*, 2019. arXiv:1908.00063.

32 Xiaoyin Ge, Issam I. Safa, Mikhail Belkin, and Yusu Wang. Data skeletonization via Reeb graphs. In J. Shawe-Taylor, R.S. Zemel, P. Bartlett, F.C.N. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 837–845, 2011.

33 William Harvey and Yusu Wang. Topological landscape ensembles for visualization of scalar-valued functions. In *Proceedings of the 12th Eurographics / IEEE - VGTC Conference on Visualization*, EuroVis'10, pages 993–1002, Aire-la-Ville, Switzerland, Switzerland, 2010. Eurographics Association. doi:10.1111/j.1467-8659.2009.01706.x.

34 Franck Hétroy and Dominique Attali. Topological quadrangulations of closed triangulated surfaces using the Reeb graph. *Graphical Models*, 65(1-3):131–148, 2003. doi:10.1016/s1524-0703(03)00005-5.

35 Masaki Hilaga, Yoshihisa Shinagawa, Taku Kohmura, and Tosiyasu L. Kunii. Topology matching for fully automatic similarity estimation of 3D shapes. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pages 203–212, New York, NY, USA, 2001. ACM. doi:10.1145/383259.383282.

36 Woojin Kim and Facundo Memoli. Stable signatures for dynamic metric spaces via zigzag persistent homology. *CoRR*, 2017. arXiv:1712.04064v1.

37 Woojin Kim, Facundo Mémoli, and Anastasios Stefanou. The metric structure of the formigram interleaving distance. *CoRR*, 2019. arXiv:1912.04366v1.

38 Michael Lesnick. The theory of the interleaving distance on multidimensional persistence modules. *Foundations of Computational Mathematics*, 15(3):613–650, 2015. doi:10.1007/s10208-015-9255-y.

39 Dmitriy Morozov, Kenes Beketayev, and Gunther Weber. Interleaving distance between merge trees. In *Proceedings of TopoInVis*, 2013.

40 Elizabeth Munch and Anastasios Stefanou. The ℓ ∞-cophenetic metric for phylogenetic trees as an interleaving distance. In *Association for Women in Mathematics Series*, pages 109–127. Springer International Publishing, 2019. doi:10.1007/978-3-030-11566-1_5.

41 Elizabeth Munch and Bei Wang. Convergence between categorical representations of Reeb space and Mapper. In Sándor Fekete and Anna Lubiw, editors, *32nd International Symposium on Computational Geometry (SoCG 2016)*, volume 51 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 53:1–53:16, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.SoCG.2016.53.

42 Mattia Natali, Silvia Biasotti, Giuseppe Patanè, and Bianca Falcidieno. Graph-based representations of point clouds. *Graphical Models*, 73(5):151–164, 2011. doi:10.1016/j.gmod.2011.03.002.

43 Steve Y. Oudot. *Persistence Theory: From Quiver Representations to Data Analysis (Mathematical Surveys and Monographs)*. American Mathematical Society, 2017.

44 Georges Reeb. Sur les points singuliers d'une forme de pfaff complèment intégrable ou d'une fonction numérique. *Comptes Rendus de L'Académie ses Séances*, 222:847–849, 1946.

45 Luis Scoccola. *Locally persistent categories andmetric properties of interleaving distances*. PhD thesis, Western University, 2020.

46 Gurjeet Singh, Facundo Mémoli, and Gunnar Carlsson. Topological methods for the analysis of high dimensional data sets and 3D object recognition. In *Eurographics Symposium on Point-Based Graphics*, 2007.

47 Raghavendra Sridharamurthy, Talha Bin Masood, Adhitya Kamakshidasan, and Vijay Natarajan. Edit distance between merge trees. *IEEE Transactions on Visualization and Computer Graphics*, pages 1–1, 2018. doi:10.1109/tvcg.2018.2873612.

48 Anastasios Stefanou. *Dynamics on Categories and Applications*. PhD thesis, University at Albany, State University of New York, 2018.

49 Anastasios Stefanou. Tree decomposition of reeb graphs, parametrized complexity, and applications to phylogenetics. *Journal of Applied and Computational Topology*, February 2020. doi:10.1007/s41468-020-00051-1.

50 Andrzej Szymczak. A categorical approach to contour, split and join trees with application to airway segmentation. In V. Pascucci, H. Tricoche, H. Hagen, and J. Tierny, editors, *Topological Methods in Data Analysis and Visualization*, pages 205–216. Springer, 2011.

51 G.H. Weber, P.-T. Bremer, and V. Pascucci. Topological landscapes: A terrain metaphor for scientific data. *Visualization and Computer Graphics, IEEE Transactions on*, 13(6):1416–1423, November 2007. doi:10.1109/TVCG.2007.70601.

52 Zoë Wood, Hugues Hoppe, Mathieu Desbrun, and Peter Schröder. Removing excess topology from isosurfaces. *ACM Trans. Graph.*, 23(2):190–208, 2004. doi:10.1145/990002.990007.

53 Lin Yan, Yusu Wang, Elizabeth Munch, Ellen Gasparovic, and Bei Wang. A structural average of labeled merge trees for uncertainty visualization. *IEEE Transactions on Visualization and Computer Graphics*, pages 1–1, 2019. doi:10.1109/tvcg.2019.2934242.

54 Hiroki Yuda. Topological smoothing of Reeb graphs. Master's thesis, St. Louis University, 2019.

Algorithms for Contractibility of Compressed Curves on 3-Manifold Boundaries

Erin Wolf Chambers
St. Louis University, MO, USA

Francis Lazarus
G-SCOP, CNRS, UGA, Grenoble, France

Arnaud de Mesmay
LIGM, CNRS, Université Gustave Eiffel, ESIEE Paris, 77454 Marne-la-Vallée, France

Salman Parsa
St. Louis University, MO, USA

Abstract

In this paper we prove that the problem of deciding contractibility of an arbitrary closed curve on the boundary of a 3-manifold is in **NP**. We emphasize that the manifold and the curve are both inputs to the problem. Moreover, our algorithm also works if the curve is given as a compressed word. Previously, such an algorithm was known for simple (non-compressed) curves, and, in very limited cases, for curves with self-intersections. Furthermore, our algorithm is fixed-parameter tractable in the complexity of the input 3-manifold.

As part of our proof, we obtain new polynomial-time algorithms for compressed curves on surfaces, which we believe are of independent interest. We provide a polynomial-time algorithm which, given an orientable surface and a compressed loop on the surface, computes a canonical form for the loop as a compressed word. In particular, contractibility of compressed curves on surfaces can be decided in polynomial time; prior published work considered only constant genus surfaces. More generally, we solve the following normal subgroup membership problem in polynomial time: given an arbitrary orientable surface, a compressed closed curve γ, and a collection of disjoint normal curves Δ, there is a polynomial-time algorithm to decide if γ lies in the normal subgroup generated by components of Δ in the fundamental group of the surface after attaching the curves to a basepoint.

2012 ACM Subject Classification Mathematics of computing → Geometric topology; Mathematics of computing → Graphs and surfaces; Theory of computation → Data compression

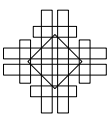
Keywords and phrases 3-manifolds, surfaces, low-dimensional topology, contractibility, compressed curves

Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.23

Related Version *Full Version*: `https://arxiv.org/pdf/2012.02352.pdf`

Funding *Erin Wolf Chambers*: This work was funded in part by the National Science Foundation through grants CCF-1614562, CCF-1907612 and DBI-1759807.
Francis Lazarus: This author is partially supported by the French ANR projects GATO (ANR-16-CE40-0009-01) and MINMAX (ANR-19-CE40-0014) and the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program Investissement d'avenir.
Arnaud de Mesmay: This author is partially supported by the French ANR projects ANR-17-CE40-0033 (SoS), ANR-16-CE40-0009-01 (GATO) and ANR-19-CE40-0014 (MINMAX).
Salman Parsa: This work was funded in part by the National Science Foundation through grant CCF-1614562 as well as funding from the SLU Research Institute.

Acknowledgements The authors would like to thank Mark Bell, Ben Burton, and Jeff Erickson for helpful discussions. We also thank an anonymous referee of [11] for suggesting the use of a maximal compression body as a simple exponential-time algorithm for deciding contractibility of arbitrary (non-compressed) curves on the boundary of a 3-manifold.



1 Introduction

In a topological space X, a closed curve $c : \mathbb{S}^1 \to X$ is *contractible* if the map c extends to a map of the standard disk, $f : D \to X$, $f|_{\partial D} = c$; such a curve can be continuously deformed to a point. Dating back to Dehn [12] more than a century ago, the problem of testing contractibility has been an important focus in the development of efficient algorithms in computational topology and group theory. Such problems are now well understood in two dimensions, with recent works of the second author and Rivaud [29] and Erickson and Whittlesey [17] providing linear-time algorithms to test contractibility and free homotopy of curves on surfaces; Despré and the second author [13] also developed efficient algorithms for the closely related problem of computing the geometric intersection number of curves. On the other hand, in higher dimensions, homotopy problems quickly become undecidable: testing contractibility of curves is already undecidable on 2-dimensional simplicial complexes and in 4-manifolds (see Stillwell [39, Chapter 8]).

The aim of this paper is to further improve our understanding of the intermediate 3-dimensional case. In a 3-manifold, testing whether a curve is contractible is known to be decidable, but the best known algorithms are intricate and rely on the proof of the Geometrization Conjecture. We refer to the survey of Aschenbrenner, Friedl and Wilton [2]. When the input manifold is a knot complement, this problem contains as a special case the famous Unknot Recognition problem, which asks whether an input knot in $\mathbb{R}^3$ is trivial. That problem is now known to be in **NP** [19] and co-**NP** [26]. Recently, Colin de Verdière and the fourth author [10] initiated the study of a problem in between those two, which is testing the contractibility of a closed curve γ which lies on the boundary of a triangulated 3-manifold M, and they provided an algorithm to solve this problem in time $2^{O(|M|+|\gamma|)^2}$). They asked whether the problem lies in the complexity class **NP** and proved that it holds in two cases: when the number of self-intersections of γ is at most logarithmic, and when the boundary surface is a torus. We remark that the existence of an algorithm to test contractibility of a closed curve on the boundary actually follows from the earlier results of Waldhausen [40], and an exponential-time algorithm can be derived from the standard 3-manifold literature, as we will explain later in this paper. However, the algorithm of [10, 11] is simpler in the sense that it relies only on the proof of the Loop Theorem.

Our main result is that the problem of contractibility when the curve is on the boundary of the 3-manifold is in **NP**. Furthermore, the algorithm that we provide has two additional features. First, it is fixed parameter tractable in the input manifold: for a given 3-manifold M, after a preprocessing taking exponential time in $|M|$, we can solve any contractibility test for curves on the boundary in polynomial time. Second, our algorithm also works if the input curve is *compressed* via a straight-line program, in particular, it can be exponentially longer than its encoding size (we defer to Section 2 for the precise definitions).

▶ **Theorem 1.** *Let a triangulated 3-manifold M and a curve γ on the boundary of M be given as the input. The problem of deciding whether γ is contractible in M is in **NP**, and there is a deterministic algorithm that solves the problem in time $2^{O(|M|^3)}poly(|\gamma|)$. This is also the case if γ is given as a compressed word, i.e., a straight-line program.*

Our proof of Theorem 1 will reduce the problem to another problem involving only curves on surfaces. However, there is an important subtlety. Many algorithms in 3-dimensional topology involve an exponential blow-up in the size of the objects under consideration. This is the case for example for Unknot Recognition, where the classical algorithm [19] looks for a disk spanning the knot, but this disk might be exponentially large [20]. In order to

get **NP** membership, this exponential blow-up is controlled using a compressed[1] system of coordinates called *normal coordinates* (see Section 2 for definitions). Our approach follows a similar scheme and suffers from a similar blow-up, which forces us to deal with normal curves. Precisely, we show that the problem of contractibility of an arbitrary curve on the boundary of a 3-manifold reduces to the following problem. Given a family of disjoint normal closed curves Δ on a triangulated surface S, and a curve γ on S, decide if γ lies in the normal subgroup of the fundamental group determined by the curves of Δ (appropriately connected to the basepoint). This is equivalent to deciding if the curve γ is contractible in the space resulting from S by gluing a disk to each component of Δ. We call this problem the *disjoint normal subgroup membership* problem, and we provide an algorithm to solve it in polynomial time, despite the highly compressed nature of Δ and γ.

▶ **Theorem 2.** *Let an orientable triangulated surface S, a closed normal multi-curve Δ, and a compressed curve γ be given as the input. There is a polynomial-time algorithm for deciding the disjoint normal subgroup membership problem for γ and Δ on S.*

To solve the disjoint normal subgroup membership problem, one of our technical tools is a polynomial-time algorithm to test triviality of a compressed closed curve on a surface, which might be of independent interest. In fact, we compute a canonical form for such a curve, such that there is a unique canonical representative in each based homotopy class.

▶ **Theorem 3.** *Let an orientable triangulated surface S and a compressed loop γ on S described by a straight-line program be given. Then one can transform γ, in polynomial time, into a canonical form in its based homotopy class, given as a compressed word. In particular, we can test in polynomial time whether γ is contractible.*

We emphasize that in Theorem 3 the surface S is part of the input. The analogous theorem, if we assume that surface S is not part of the input, has been known in a much broader context, as we discuss in the next section.

1.1 Related work

Algorithms in 3-manifold topology. There is now a large body of research on the computational complexity of problems in 3-manifold topology and knot theory. To paint a picture in broad strokes, most topological problems are now known to be decidable, using a combination of geometric, algebraic and topological tools and the complexity of the best known algorithms range from exponential [19, 1] to quite galactic, see for example Kuperberg [25] for the homeomorphism problem. In contrast, only very few complexity lower bounds are known [1, 27, 9] and for many problems, including ours, it is open whether a polynomial-time algorithm exists. One particular tool that our work relies on is *normal surface theory*, which provides a powerful framework to enumerate and analyze the topologically relevant embedded surfaces in a 3-manifold. Actually, tools from normal surface theory [38] provide an off-the-shelf algorithm proving that deciding contractibility of a *simple* curve on the boundary of a 3-manifold is in **NP**, as explained in [10, Section 3]. However, normal surface theory is ill-adapted to study surfaces that are not embedded [5], hence the more technical path that we follow in this paper. We refer to Lackenby [28] for a recent and extensive survey on algorithms in 3-manifold topology.

[1] There are two different forms of compression going on in this paper: normal coordinates and straight line programs. We keep the name *compressed* for the latter, while a curve or a multicurve encoded with normal coordinates will be simply called a *normal (multi-)curve*.

Geometric and combinatorial group theory. The problem of deciding the contractibility of a curve in a topological space, which is given for example as a finite simplicial complex, can be equivalently rephrased as a problem of testing the triviality of a word in a finitely presented group, the fundamental group of the space. In the case of 2- and 3-manifolds, the geometry or topology of the underlying manifold has a strong impact on the properties of the group, and there is a vast amount of literature on this interaction, see for example [8]. We refer to Aschenbrenner, Friedl and Wilton [2, 3] for the case of 3-manifold groups. Of particular interest for the word problem is the notion of a *Dehn function*, which upper bounds the area of a disk certifying the triviality of a word, and thus controls the complexity of a brute-force algorithm to solve the word problem. For 3-manifold groups, this function can be exponential [7, Theorem 8.1.3], making such an approach at least exponentially worse than ours. When the Dehn function of a group presentation is polynomial (as is the case, for instance, for the fundamental group of a compression body), it can be used to prove that the word problem is in **NP**. However, the word problem is defined for a fixed presentation. In our problem, the group presentation itself is part of the input, and we cannot use the bound on the Dehn function directly.

Algorithms for compressed curves and surfaces. As we hinted at in the introduction, 3-dimensional algorithms naturally tend to involve exponentially large objects, which are generally compressed using normal coordinates. This incurs a need for efficient algorithms dealing with normal curves and surfaces, even for the most basic tasks. For example, it is not easy to test whether a curve described by normal coordinates is connected. By now, there are many known techniques to handle topological problems on compressed curves, see for example Agol, Hass and Thurston [1], Schaefer, Sedgwick and Stefankovič [35, 36], Erickson and Nayyeri [16], Bell [4] or Dynnikov [14] and Dynnikov and Wiest [15]. None of these techniques seem to directly solve our disjoint normal subgroup membership problem, but we rely extensively on the work of Erickson and Nayyeri [16] as a subroutine (see Section 4). However, normal coordinates do not describe curves with self-intersections, which are our main object of interest in this paper. This is why we rely on a different compression model in the form of *straight-line programs*. The use of such programs in geometric group theory is not new, and in particular there is a sketch of an algorithm to test triviality of compressed curves on a surface of genus 3 in an appendix of Schleimer [37, Appendix A]. His approach seems to be readily generalizable to higher genus, but the dependency on the genus is not made explicit. More recently, Holt, Lohrey and Schleimer [23] provided a polynomial-time algorithm to test triviality of compressed words in *hyperbolic groups*, of which (most) surface groups are a subclass. The difference with our Theorem 3 is that in their result, the group presentation is not part of the input. While both approaches could plausibly be used in our setting, we rely on different tools to provide a complete proof of Theorem 3: our approach treats the surface as an input and works for any genus, and seamlessly generalizes to the case of wedges of surfaces that we also need.

1.2 Summary of our techniques

The standard methods of normal surface theory allow us to reduce our main contractibility problem to the case where the input manifold belongs to a particular class of manifolds called *compression bodies*, see Section 3. The fundamental group of a compression body is roughly a free product of surface groups, and thus one can rely on known algorithms for surfaces [17, 29] to solve the contractibility problem there. However, due to the exponential size of normal surfaces, this would only yield an exponential time algorithm. In order to

improve on this, we identify the precise problem that we need to solve to be the disjoint normal subgroup membership problem and set out on a quest to solve it efficiently. While an **NP** algorithm would suffice for our main result, we will develop a polynomial-time algorithm.

The disjoint normal subgroup membership problem is made delicate by the compressed nature of the curves in Δ. If they were given as disjoint curves, say, on the 1-skeleton of the surface, we could glue a disk on each of them, and observe that the resulting complex is homotopically equivalent to a wedge of surfaces[2], allowing us to reduce the problem to the triviality of a curve in a surface (see Section 5 for a description of the homotopy equivalence). In order to do so despite the compression, we compute in Section 4 a *retriangulation* of the surface, so that the curves in Δ are only polynomially long in the new triangulation. This is done by tracing the normal curve and building the *street complex* which was specifically designed by Erickson and Nayyeri to handle normal curves on surfaces in polynomial time. We then track how the street complex interacts with our input curve γ. This operation comes at a cost: even if the input curve γ was not compressed, it may become exponentially long after the retriangulation. Since it may be not simple, we rely on straight-line programs instead of normal curves to encode it. At this stage, we note that it might as well have been a compressed straight-line program from the start.

Finally, we want to test the triviality of a compressed curve in a wedge of surfaces. The fundamental group of this wedge is a free product of fundamental groups of surfaces, and thus the crux of the problem is to solve it in a single surface. While there are known techniques to test the contractibility of a curve on a surface, based on local simplification rules [17, 29], the compressed nature of our curves is once again a stumbling block here, as we need to be careful to never apply an exponential number of such simplification rules. The solution sketched by Schleimer [37, Appendix A] to that issue is to introduce a family of *well-tempered paths*, which are carefully designed to not necessitate such exponential simplifications. We use a different approach: we rely on the standard tools developed for the non-compressed case, namely quad systems and turn sequences, and detect exponential simplification and realize them all at once in polynomial time. This relies on recent insights on the structure of these quad systems [13] [30, Section 4.3].

Most proofs are omitted due to space constraints; complete details are provided in the full version, which is appended after this extended abstract.

2 Background

We begin by recalling some key definitions and results, although we assume that the reader is familiar with basic algebraic topology such as homotopy and fundamental groups; we refer to Hatcher [21] or Stillwell [39] for more detailed definitions.

2.1 3-Dimensional manifolds

A 3-manifold is a topological space that is locally homeomorphic to a 3-dimensional ball or a 3-dimensional half-ball. The points of the latter type form the *boundary* of the 3-manifold, which is always a surface. In this paper, 3-manifolds are always assumed to be triangulated, and we use common and a less restrictive definition than simplicial complexes: a *triangulation* of a 3-manifold M is a collection of n abstract tetrahedra, along with a collection of gluing

[2] A *wedge* of a family topological spaces is the space obtained after attaching them all to a single common point. Actually, there are also circle summands here – we do not mention them in this outline to keep the discussion light.

pairs which identify some of the $4n$ boundary triangles in pairs. In particular, we allow two faces of the same tetrahedron to be identified. If we add the further restriction that the neighborhood of each vertex is a 3-ball or a half 3-ball and no edge gets glued to itself with the opposite orientation, then the resulting space is always a 3-manifold [34].

This paper makes heavy use of normal surface theory; see the full version for more definitions and relevant background on this topic.

2.2 Curve and surface representations

All the surfaces in this article are closed, i.e., without boundary. We shall often represent a surface by a graph cellularly embedded in that surface. This embedding is encoded as a *combinatorial* surface thanks to a rotation system over the edges of the graph [33]. Equivalently, one can define a combinatorial surface as a gluing of polygons whose sides are pairwise identified. When all the polygons are triangles, we speak of a *triangulation*, or a *triangulated surface*. Note that we allow a triangle to have two sides identified after the gluing or two triangles to share two vertices but no edge, etc.

The *complexity* of a surface is the number of cells (vertices, edges and faces) of the complex induced by the graph embedding. A curve on such a surface can be represented or encoded in a variety of ways. For instance, any curve is homotopic to a curve defined by a walk on the 1-skeleton of the complex. The complexity of the curve is the number of edges in the walk counted with repetition. One can also consider curves in general position avoiding the vertices of the graph and cutting its edges transversely as in the next section, in which case we say the complexity of a curve is its number of intersection points with the graph.

We next outline two different notions of compressed curve representations. To repeat the footnote of warning in the introduction: normal coordinates and straight line programs are two different methods for compressed representations, both of which we use in this paper. We keep the name *compressed* for straight line programs, while a curve or a multicurve encoded with normal coordinates will be simply called a *normal (multi-)curve*.

Normal curves. Let S be a triangulated surface. A *multi-curve* is a disjoint family of curves embedded on a S. A multi-curve c is *normal* if it is in general position with respect to the triangulation of S and if every maximal arc of c in a triangle has its endpoints on distinct sides. In particular, no component of a normal curve is contained in the interior of a single triangle. A normal curve may have three types of arcs in a triangle, one for each pair of sides. Counting the number of arcs of c of each type in each of the t triangles of the triangulation, we get $3t$ numbers called the *normal coordinates* of c. A *normal isotopy* of S is an isotopy that leaves each cell of the triangulation invariant. Up to a normal isotopy, c is completely determined by its normal coordinates. It is thus possible to encode a multi-curve with exponentially many arcs by storing its normal coordinates, which then have polynomial bit-complexity. A normal multi-curve is *reduced* if no two of its components are normally isotopic.

Straight-line programs. Another method of compressing curves is to think of a curve as a word over an alphabet. In this encoding, a letter of the alphabet corresponds to a directed edge on the surface and juxtaposition of letters corresponds to concatenation of paths. Therefore, we can consider curves as abstract words and to represent them using compressed representations of abstract words. The length of a word w, denoted by $|w|$, is the number of letters in it.

A *straight-line program* is a four-tuple $\mathbb{A} = \langle \mathcal{L}, \mathcal{A}, A_n, \mathcal{P} \rangle$ where:

- $\mathcal{L}$ is a finite alphabet of *terminal* characters,
- $\mathcal{A}$ is a disjoint alphabet of *nonterminal* characters,
- $A_n \in \mathcal{A}$ is the *root*, and
- $\mathcal{P} = \{A_i \to W_i\}$ is a sequence of *production rules*, where W_i is a word in $(\mathcal{L} \cup \mathcal{A})^*$ containing only non-terminals A_j for $j < i$.

A straight-line program is in *Chomsky normal form* if every production rule either has two non-terminals or a single terminal on the right. Every straight-line program can be put in polynomial time into Chomsky normal form by adding intermediate non-terminals, and thus we will always assume that a straight-line program is in Chomsky normal form. We denote by $w(A)$ the word encoded by a non-terminal character A, or sometimes simply by A when there is no confusion. Its length is denoted by $|A|$. We will always use a terminal alphabet $\mathcal{L}$ equipped with an involution $a \mapsto \bar{a}$. The *reversal* of a word w is the word obtained by changing its letter a by its reverse $\bar{a}$ and reversing the order of the letters.

Composition systems are straight-line programs of a more advanced type, where productions of the form $P \to A[i:j]B[k:l]$ are allowed, and the meaning is that $A[i:j]$ represents the subword between the $i+1$th and jth letter (both included, starting the numbering at 1) of the word that A encodes. We take the convention that negative indices count from the end of the word, and that $A[i:]$, $A[i]$ and $A[:j]$ are shorthand respectively for $A[i:-1]$, $A[i:i+1]$ and $A[0:j]$. While composition systems might seem more powerful than straight-line programs, a theorem of Hagenah [18] says that given any composition system, one can compute in polynomial time an equivalent straight-line program. Henceforth, we will slightly abuse language and freely use the $[\cdot:\cdot]$ construct in our straight-line programs.

The following theorem summarizes the algorithms on straight-line programs that we will rely on, see Schleimer [37] or Lohrey [32].

▶ **Lemma 4.** *Let $\mathbb{A}$ and $\mathbb{X}$ be two straight-line programs, i be an integer and e be a letter in the terminal alphabet of $\mathbb{A}$. Then one can, in polynomial-time,*

- *compute the length of $\mathbb{A}$,*
- *output the letter $\mathbb{A}[i]$,*
- *find the greatest j so that $\mathbb{A}[j] = e$,*
- *compute a straight-line program $\overline{\mathbb{A}}$ for the reversal word $\overline{w(\mathbb{A})}$,*
- *decide whether $w(\mathbb{A}) = w(\mathbb{X})$,*
- *find the biggest k such that $\mathbb{A}[:k] = \mathbb{X}[:k]$.*

For a given 2-complex K (in particular if K is a combinatorial surface), a straight-line program with terminal alphabet $\mathcal{L}$ the set of directed edges of K can encode any closed curve on K. We call such an encoding a *compressed curve* or *walk*. Simple operations on K can seamlessly be done while updating a compressed curve appropriately to obtain a homotopic compressed curve in the modified complex. For example, contracting an edge e of K simply boils down to adding a production rule $e \to \varepsilon$, where ε is the empty word. Likewise, deleting an edge e bounding a face $\bar{e}p$, where p is the complementary path in that face, amounts to replacing each occurrence of e by a p via a new production rule $e \to p$. When no encoding is specified for a closed (multi-)curve, then it is simply encoded as a (multi-)walk on the one-skeleton of the underlying surface or complex.

3 From contractibility to normal subgroup membership problem

The following proposition reduces the contractibility problem for closed curves on the boundary of an orientable 3-manifold M to the normal subgroup membership problem for a boundary surface of M, where the multi-curve Δ is a reduced normal curve with polynomial complexity. We note that this reduction is the only place where the algorithm is non-deterministic, the rest of our algorithms run in deterministic polynomial time.

▶ **Proposition 5.** *Let M be an orientable triangulated 3-manifold with boundary. There exists a reduced normal multi-curve Δ on ∂M with normal coordinates of linear bit-size (with respect to $|M|$), so that for any curve γ on ∂M, γ is contractible in M if and only if γ is contained in the normal subgroup N of $\pi_1(\partial M)$ generated by the homotopy classes of the components of Δ.*

Further, given M, Δ, and a cubic-sized certificate (in $|M|$), there is a polynomial-time algorithm to verify that Δ has the property that all curves in N are contractible in M.

Our proof begins by crushing M to get an irreducible manifold [6] with the same triangulated boundary, and then relies on the fact that normal surface theory [24] proves the existence of a complete family of *compression disks*, whose boundary is the reduced multi-curve Δ. Its properties follow from the Loop Theorem [22, Theorem 4.2]. We refer to the full version for details.

Note that we did not strive to optimize the size of the certificate in this proposition, since it does not matter for our application, so further improvements may be possible. In particular, using enumeration techniques for vertex surfaces [19, Section 6], we suspect it can be shown to be quadratic rather than cubic.

▶ **Proposition 6.** *The problem of contractibility of a compressed curve on the boundary of an arbitrary 3-manifold reduces, in polynomial time, to the problem of contractibility of a compressed curve on the boundary of an orientable 3-manifold.*

Relying on Proposition 6, for the rest of the paper we restrict our attention to orientable surfaces.

Hardness of the general normal subgroup membership problem. If the curves Δ in the normal subgroup membership problem are not disjoint then this problem is much harder, and possibly undecidable, even if they have few edges, see the full version for details.

4 Retriangulation

Given a reduced normal multi-curve Δ on a triangulated 2-manifold, in this section we present a way to compute a new triangulation of polynomial complexity, in which the curves of Δ lie in the 1-skeleton and are only polynomially long.

▶ **Proposition 7.** *Let a simplicial complex K triangulating a surface S, a reduced normal multi-curve Δ and a compressed closed walk γ on the 1-skeleton of K be given as input. The size of the input is the summation of the complexities of the curves γ, Δ and complexity of K. There is an algorithm that in polynomial time computes a new simplicial complex K' triangulating the same surface S, a multi-curve Δ', and a walk γ' on the 1-skeleton of K' given as a composition system, such that:*

- *there is a homeomorphism $f : |K| \to |K'|$, such that the images of Δ under f coincides with Δ',*
- *γ' is homotopic to $f(\gamma)$,*
- *the multi-curve Δ' is an embedded multi-curve on the 1-skeleton of K'.*

Consequently, the homotopy class of γ' belongs to the normal subgroup generated by the components of Δ' in $|K'|$ if and only if the homotopy class of γ belongs to the normal subgroup generated by the components of Δ in $|K|$.

Note that in this proposition, the complexities of the inputs γ and Δ are compressed.

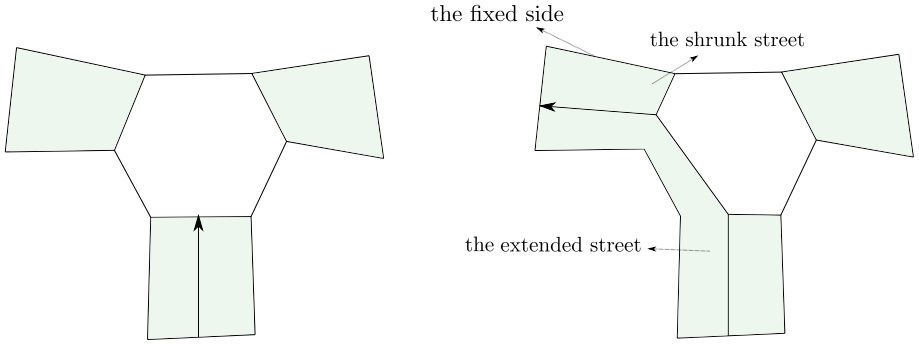


Figure 1 A left turn in the street complex; the arrows depict segments of the edge e.

Sketch of proof. We briefly outline our approach, and again refer the reader to the full version for details. Our algorithm relies heavily on the street complex introduced by Erickson and Nayyeri [16] (see also Bell [4]); we briefly recall its structure here. The overlay of Δ with K cuts the triangles of K into smaller pieces. We abuse notation and call quadrilaterals the pieces bounded by exactly two subedges of K, though they can actually be degenerate triangles. Merging the quadrilaterals whenever they share an edge segment of K results in the *street complex*. The maximal sequences of merged quadrilaterals are called *streets*, and the remaining faces are *junctions*. Quite surprisingly, the size of the street complex is bounded by a linear function of the complexity of K [16, Lem. 2.3], independent of Δ. Note that the curves in Δ appears in the 1-skeleton of this complex at the street boundaries. The complex K' in the proposition is essentially this street complex, further triangulated, and Δ' is the image of Δ in K'.

In [16] the algorithm to construct the street complex works inductively by *tracing* the curves in Δ. Intuitively, the tracing algorithm follows each curve from an arbitrary chosen point and extends one of the incident streets as the curve traverses the current streets and junctions. See Figure 1. An essential step in our algorithm is to somehow reverse the roles of K and Δ to compute for each edge e of K a composition system $\mathbb{E}$ encoding its intersections with Δ. The terminals of this system correspond to the segments of e cut by Δ. We compute $\mathbb{E}$ inductively in parallel to the street complex construction, following the tracing algorithm of Erickson and Nayyeri. Each terminal of $\mathbb{E}$ is contained in some street and can be homotoped to a path in the boundary of this street. We replace the terminal by this path to obtain a composition system encoding a curve homotopic to e, while contained in K'. At a very final step, we replace the terminal edges of the straight line program for γ by the corresponding composition systems. Our algorithm can be executed in polynomial time, by adapting the analysis of Erickson and Nayyeri [16]. ◀

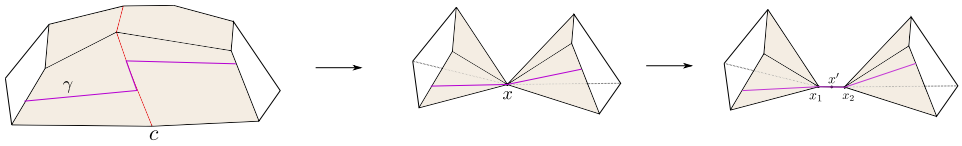


Figure 2 Contraction of a component c of Δ (shown on the left) results in a new complex K' (shown on the right).

5 Capping off

In this section, we show that the complex obtained by gluing disks on a family of disjoint curves on a surface is homotopy equivalent to a wedge of surfaces and circles. We provide a polynomial-time algorithm to compute the resulting wedge, while also tracking what happens to a compressed curve on the original surface.

▶ **Proposition 8.** *Let L be simplicial complex triangulating a surface S of genus ≥ 1, Δ be a family of disjoint embedded closed curves in the 1-skeleton of L and γ be a compressed closed walk on the 1-skeleton of L. The size of the input is the summation of the complexities of L, γ and the number of edges of Δ. We can compute in polynomial time a simplicial complex K which is a wedge of surfaces and circles, and a compressed walk w on K, so that:*

- *K is homotopy equivalent to the complex obtained by gluing disks on each component of Δ,*
- *w is homotopic to a trivial walk in K if and only if γ belongs to the normal subgroup determined by the curves in Δ.*

The proof is deferred to the full version, but the key idea is illustrated in Figure 2: we repeatedly pinch the curves in Δ to a single point, extend this point to a small path and contract a spanning tree in the rest to have a single vertex, and thus a wedge.

6 Triviality for compressed words in free products of surface groups

In this section, we prove the following theorem:

▶ **Theorem 9.** *Let K be a wedge of combinatorial surfaces and circles and w be a compressed walk on K. Then one can compute in polynomial time a canonical form for the homotopy class of w. In particular, we can test in polynomial time whether w is trivial.*

We emphasize that in this theorem, we consider w as a walk with fixed endpoints, and thus we are considering based homotopies (as opposed to free homotopies).

The proof of Theorem 9 is rather involved, and we only outline the main ideas here, in the simpler case where the wedge consists of a single surface of negative Euler characteristic, omitting proofs of the intermediate lemmas. Since the fundamental group of a wedge of spaces is the free product of the fundamental groups of the spaces, the more general version is not much harder. We refer to the full version for complete details.

Our algorithm starts in a similar way as the known linear-time algorithms to test contractibility or homotopy of curves on surfaces [17, 29]: we first turn our surface into a system of quads (Lemma 10), and then compute a canonical form for our curve w (Lemma 13). This canonical form is unique, and therefore the walk w is homotopic to a trivial walk if and only if its reduced canonical form is the empty walk. However, due to the compression of the input, our techniques to reduce the curve w are more involved than those of the aforementioned references.

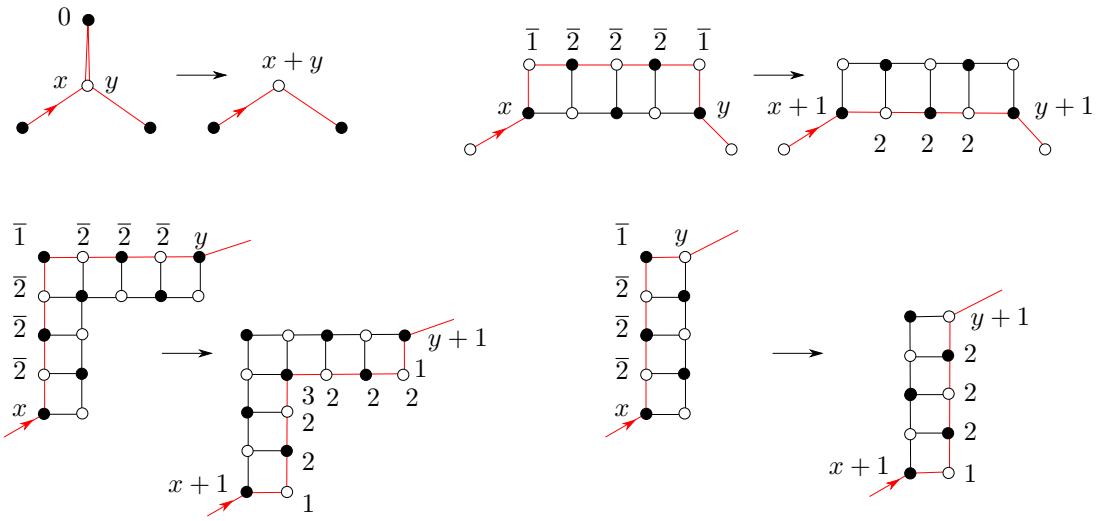


Figure 3 The reductions to eliminate a spur (top left) and a bracket (top right), and the shifting moves to remove a $\bar{1}$ (bottom row).

Our first step is to turn our surface into systems of quads. First, we compute a spanning tree and contract it. Then, we remove edges until there is a single face, we add a new vertex inside the single face, add all the radial edges between this new vertex and the single vertex of S and remove all the previous edges. The resulting complex is called a *quad system*.

▶ **Lemma 10.** *Let K be a surface and let w be a compressed walk on K. In polynomial time we can turn K into a quad system K' and compute a compressed walk w' on K' that is homotopic to w.*

In the next step of our algorithm, we encode words using *turn sequences*. For any two directed edges e and e' on the same surface S, the turn $\tau(e, e')$ is the number of corners, counted with respect to our chosen orientation, between the end of e and the start of e' on S. The turn sequence of our word is the concatenation of all the turn sequences of consecutive edges. However, we lose information by encoding with turn sequences. First, a turn sequence does not specify a starting edge. Second, even if we know the starting edge, it is not immediately clear how to compute an arbitrary intermediate edge along the path efficiently since we encode exponential length paths. The following lemma shows how to do that in polynomial time.

▶ **Lemma 11.** *Let $\mathbb{T}$ be a compressed turn sequence and k be an integer. Then if we are given a starting edge $e_{initial}$, then in polynomial time we can compute the edge of K we are on just after starting at $e_{initial}$ and following the turn sequence up to $\mathbb{T}[k]$.*

The point of turn sequences is that they make it easier to make local simplifications to a contractible curve. Following Lazarus-Rivaud [29] and Erickson-Whittlesey [17], we use the following simplification rules, see Figure 3, which are homotopies.

- A *spur* in a turn sequence w is a 0 turn. Removing a spur is applying the rule $x0y \to x+y$.
- A *bracket* in a turn sequence w is a subword of the form $12\dots21$ or $\bar{1}\bar{2}\dots\bar{2}\bar{1}$. Flattening a bracket is applying the rules $x12\dots21y \to (x-1)\bar{2}\dots\bar{2}(y-1)$ or $x\bar{1}\bar{2}\dots\bar{2}\bar{1}y \to (x+1)2\dots2(y+1)$.

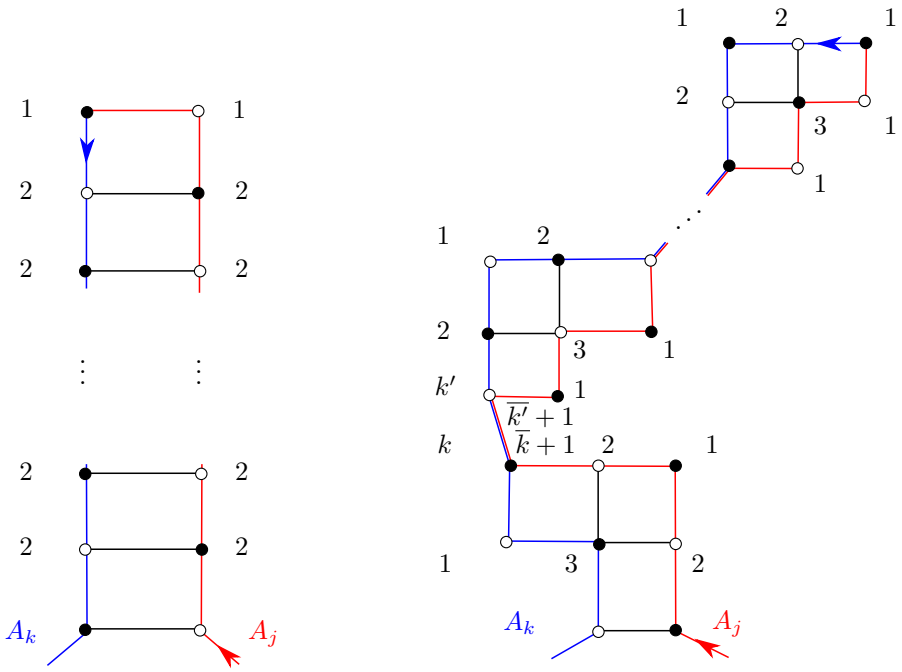


Figure 4 Some production rules $A_i \to A_j A_k$ require an exponential number of bracket flattenings and/or spur removals to be reduced, despite A_j and A_k being already reduced.

We are only considering homotopies of paths in this paper, and thus do not need the other special cases considered in Erickson-Whittlesey [17]. Our goal is to modify the turn sequence so that it is *reduced*, i.e., contains neither spurs nor brackets. However, unlike in the aforementioned works, we cannot apply these rules directly, even inductively: in a production rule $A_i \to A_j A_k$, even when A_j and A_k are reduced, there might be an exponential number of bracket flattenings in A_i, for example in the cases in Figure 4.

There are known techniques to handle exponentially long spurs [31, 37], but for the more intricate cases pictured in Figure 4, especially the kind on the right side, we need to develop our own tools. In order to do that, we rely on stronger inductive forms, similarly to those used in the free homotopy test [17, 29]. A reduced path is *leftmost* (or *rightmost*, respectively) if its turn sequence contains no 1, respectively no $\bar{1}$. One can transform a reduced path into its rightmost (resp. leftmost) form by doing *elementary right-shifts* (resp. *elementary left-shifts*); see Figure 3, bottom. Rightmost and leftmost paths are unique in their homotopy classes [17, 29].

For each character in the straight-line program, we inductively compute both a rightmost and leftmost turn sequence. Then, both are used to carry the induction step. So the next step of our algorithm is the following reduction algorithm. Since turn sequences only encode the turns at the interior vertices of a path and forget where the path starts, we store this information separately: for each character in the straight-line program, we store in a dictionary its starting and ending edges (which might be empty). To keep the description concise, we explain the algorithm in words, and refer to the full version for a much more detailed description.

Algorithm 1 Reduction Algorithm.

Input: A compressed walk on a quad system, as output by Lemma 10.
Output: Two compressed turn sequences on a quad system which are the leftmost and rightmost forms of the input, and their starting and ending edges.

- A leaf of the production tree is a single edge, which is turned to ε, and the edge is stored in the dictionary.
- Let $A_i \to A_j A_k$ be a production rule, and assume that we have inductively computed rightmost and leftmost reduced turn sequences for A_j and A_k, which are denoted by A_j^R, A_j^L, A_k^R and A_k^L, as well as their starting and ending edges. We explain how to compute a rightmost reduced turn sequence A_i^R, the leftmost case being symmetric.
  1. Using Lemma 4 and the dictionary, we compute a maximal common path between the end of A_k^R and the start of A_k^L (let us emphasize that we use the **leftmost** form here). We trim the end of A_j^R and the start of A_k^R by the length of this path, manually reconnecting them if there is some offset.
  2. Using Lemma 4, we remove a maximal spur or ladder between A_j^R and A_k^R (as on the left of Figure 4).
  3. At this stage, no bad cases can happen anymore. We simply perform a single elementary right shift, remove the at most two remaining brackets, and update the starting and ending edge in the dictionary.

Step 3 simply implements the bracket flattenings and the shifts, so the main mysteries in this algorithm are steps 1 and 2. These are tailored to deal with the bad cases pictured in Figure 4, which are the only possible bad cases; see the proof of Lemma 12. Figure 5 illustrates how step 1 reduces (a subset of) the right case of Figure 4 to its left case, and how step 2 deals with that case. Note that in these pictures, A_j and A_k were already rightmost.

We claim that after this procedure, the path that represents the word A_i^R (respectively A_i^L) is homotopic to the path represented by A_i, and is rightmost (respectively leftmost). The following lemma is the crux of the analysis. We state it for rightmost paths but the symmetric version with leftmost paths holds with the same proof.

▶ **Lemma 12.** *At the end of step 2, the paths represented by A_j^R and A_k^R are rightmost. Furthermore, denoting by τ the turn between A_j^R and A_k^R at the end of step 2, the path represented by $A_j^R \tau A_k^R$ is homotopic to the one before these two steps, contains no spurs, contains at most two brackets, and these brackets contain both at least one* 2.

The idea of the proof is that due to structural results on reduced paths on quad systems [13, Theorem 10], paths that would incur long sequences of simplifications necessarily look like a sequence of paths and staircases, as in the right side of Figure 4. These are dealt with a single move in step 1, observing that taking the **leftmost** form of A_k collapses these staircases into a long spur, which we know how to handle. After this step, we prove that there is either a ladder or a spur, which are removed in step 2.

▶ **Lemma 13.** *The Reduction Algorithm runs in polynomial time. The straight-line program that it outputs has the property that every pair of characters A_i^R and A_i^L encode a pair of turn sequence corresponding respectively to the rightmost and leftmost paths homotopic to A_i. In particular, at the top level, $\mathbb{A}^R$ and $\mathbb{A}^L$ are the rightmost and leftmost paths homotopic to the compressed input walk.*

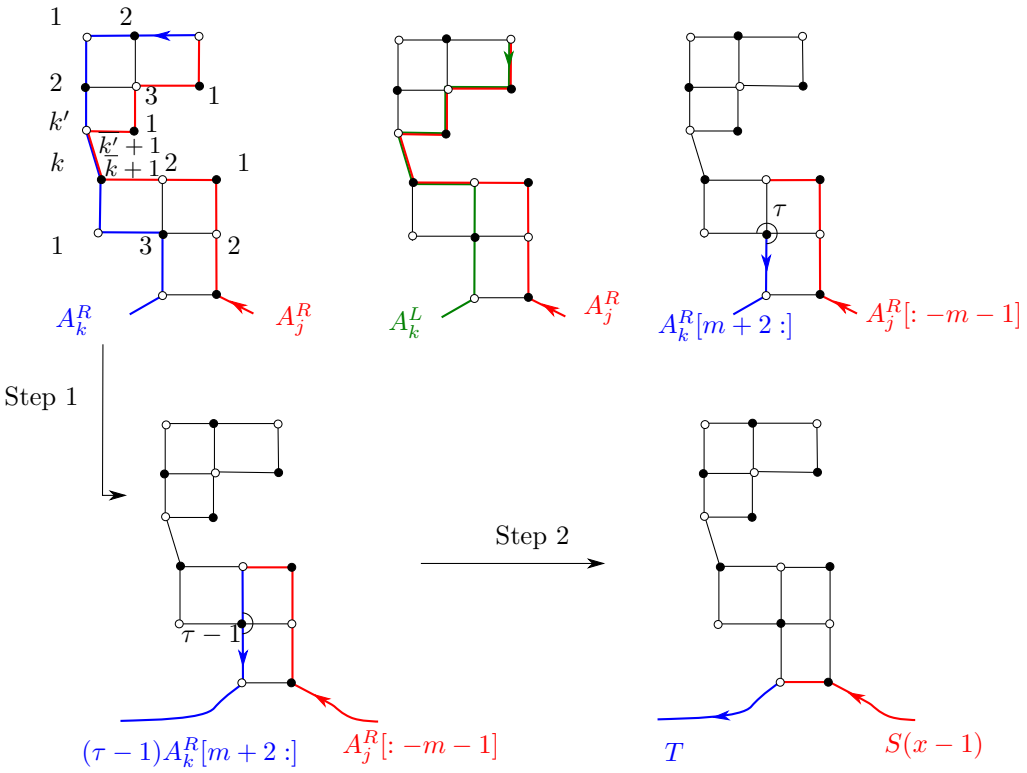


Figure 5 The leftmost form A_k^L forms a long spur with A_j^R. After reducing A_k^R by the length of this spur, all of the staircases get removed, except possibly the last one. It disappears in step 2.

Since rightmost paths are unique, the reduction algorithm applied to the root of the straight-line program produces a unique representative, and testing triviality amounts to testing emptiness. This proves Theorem 9 (see the full version for details).

Proofs of the main theorems

The proof of Theorems 1, 2 and 3 follow from Propositions 6, 5, 7, 8 and Theorem 9. We refer to the full version for details.

References

1 Ian Agol, Joel Hass, and William Thurston. The computational complexity of knot genus and spanning area. *Transactions of the American Mathematical Society*, 358(9):3821–3850, 2006.

2 Matthias Aschenbrenner, Stefan Friedl, and Henry Wilton. Decision problems for 3–manifolds and their fundamental groups. *Geometry & Topology Monographs*, 19(1):201–236, 2015.

3 Matthias Aschenbrenner, Stefan Friedl, Henry Wilton, and Stefan Friedl. *3-manifold groups*, volume 20. European Mathematical Society Zürich, 2015.

4 Mark C Bell. Simplifying triangulations. *arXiv preprint*, 2016. `arXiv:1604.04314`.

5 Benjamin Burton, Éric Colin de Verdière, and Arnaud de Mesmay. On the complexity of immersed normal surfaces. *Geometry & Topology*, 20(2):1061–1083, 2016.

6 Benjamin A Burton. A new approach to crushing 3-manifold triangulations. *Discrete & Computational Geometry*, 52(1):116–139, 2014.

7 James W Cannon, David BA Epstein, Derek F Holt, Silvio VF Levy, Michael S Paterson, and William P Thurston. *Word processing in groups*. CRC Press, 1992.

8 Pierre de La Harpe. *Topics in geometric group theory*. University of Chicago Press, 2000.

9 Arnaud de Mesmay, Yo'av Rieck, Eric Sedgwick, and Martin Tancer. The Unbearable Hardness of Unknotting. In Gill Barequet and Yusu Wang, editors, *35th International Symposium on Computational Geometry (SoCG 2019)*, pages 49:1–49:19, Dagstuhl, Germany, 2019.

10 Éric Colin de Verdière and Salman Parsa. Deciding contractibility of a non-simple curve on the boundary of a 3-manifold. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2691–2704. SIAM, 2017.

11 Éric Colin de Verdière and Salman Parsa. Deciding contractibility of a non-simple curve on the boundary of a 3-manifold: A computational loop theorem. *arXiv preprint*, 2020. arXiv:2001.04747.

12 Max Dehn. Transformation der Kurven auf zweiseitigen Flächen. *Mathematische Annalen*, 72(3):413–421, 1912.

13 Vincent Despré and Francis Lazarus. Computing the geometric intersection number of curves. *Journal of the ACM (JACM)*, 66(6):1–49, 2019.

14 Ivan Dynnikov. Counting intersections of normal curves. *arXiv preprint*, 2020. arXiv:2010.01638.

15 Ivan Dynnikov and Bert Wiest. On the complexity of braids. *Journal of the European Mathematical Society*, 9:801–840, 2007.

16 Jeff Erickson and Amir Nayyeri. Tracing compressed curves in triangulated surfaces. *Discrete and Computational Geometry*, 49:823–863, 2013.

17 Jeff Erickson and Kim Whittlesey. Transforming curves on surfaces redux. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 1646–1655. SIAM, 2013.

18 Christian Hagenah. Gleichungen mit regulären Randbedingungen über freien Gruppen, 2000.

19 Joel Hass, Jeffrey C Lagarias, and Nicholas Pippenger. The computational complexity of knot and link problems. *Journal of the ACM (JACM)*, 46(2):185–211, 1999.

20 Joel Hass, Jack Snoeyink, and William P Thurston. The size of spanning disks for polygonal curves. *Discrete and Computational Geometry*, 29(1):1–18, 2003.

21 A. Hatcher, Cambridge University Press, and Cornell University. Department of Mathematics. *Algebraic Topology*. Algebraic Topology. Cambridge University Press, 2002. URL: https://books.google.com/books?id=BjKs86kosqgC.

22 John Hempel. *3-Manifolds*, volume 349. American Mathematical Soc., 2004.

23 Derek Holt, Markus Lohrey, and Saul Schleimer. Compressed Decision Problems in Hyperbolic Groups. In Rolf Niedermeier and Christophe Paul, editors, *36th International Symposium on Theoretical Aspects of Computer Science (STACS 2019)*, volume 126, pages 37:1–37:16, Dagstuhl, Germany, 2019.

24 William Jaco and Jeffrey L Tollefson. Algorithms for the complete decomposition of a closed 3-manifold. *Illinois journal of mathematics*, 39(3):358–406, 1995.

25 Greg Kuperberg. Algorithmic homeomorphism of 3-manifolds as a corollary of geometrization. *Pacific Journal of Mathematics*, 301(1):189–241, 2019.

26 Marc Lackenby. The efficient certification of knottedness and Thurston norm. *arXiv preprint*, 2016. arXiv:1604.00290.

27 Marc Lackenby. Some conditionally hard problems on links and 3-manifolds. *Discrete & Computational Geometry*, 58(3):580–595, 2017.

28 Marc Lackenby. Algorithms in 3-manifold theory. *arXiv preprint*, 2020. arXiv:2002.02179.

29 Francis Lazarus and Julien Rivaud. On the homotopy test on surfaces. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, pages 440–449. IEEE, 2012.

30 Maarten Löffler, Anna Lubiw, Saul Schleimer, and Erin Moriarty Wolf Chambers. Computation in Low-Dimensional Geometry and Topology (Dagstuhl Seminar 19352). In *Dagstuhl Reports*, volume 9. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

31 Markus Lohrey. Word problems and membership problems on compressed words. *SIAM Journal on Computing*, 35(5):1210–1240, 2006.

32 Markus Lohrey. *The compressed word problem for groups*. Springer, 2014.

33 Bojan Mohar and Carsten Thomassen. *Graphs on Surfaces*. John Hopkins University Press, 2001.

34 Edwin E Moise. Affine structures in 3-manifolds: V. The Triangulation theorem and Hauptvermutung. *Annals of mathematics*, pages 96–114, 1952.

35 Marcus Schaefer, Eric Sedgwick, and Daniel Štefankovič. Algorithms for normal curves and surfaces. In *International Computing and Combinatorics Conference*, pages 370–380. Springer, 2002.

36 Marcus Schaefer, Eric Sedgwick, and Daniel Stefankovic. Computing Dehn twists and geometric intersection numbers in polynomial time. In *CCCG*, volume 20, pages 111–114. Citeseer, 2008.

37 Saul Schleimer. Polynomial-time word problems. *Commentarii Mathematici Helvetici*, 83:741–765, 2008.

38 Horst Schubert. Bestimmung der Primfaktorzerlegung von Verkettungen. *Mathematische Zeitschrift*, 76(1):116–148, 1961.

39 John Stillwell. *Classical topology and combinatorial group theory*, volume 72. Springer Science & Business Media, 2012.

40 Friedhelm Waldhausen. The word problem in fundamental groups of sufficiently large irreducible 3-manifolds. *Annals of Mathematics*, 88(2):272–280, 1968.

Faster Algorithms for Largest Empty Rectangles and Boxes

Timothy M. Chan
Department of Computer Science, University of Illinois at Urbana-Champaign, IL, USA

Abstract

We revisit a classical problem in computational geometry: finding the largest-volume axis-aligned empty box (inside a given bounding box) amidst n given points in d dimensions. Previously, the best algorithms known have running time $O(n \log^2 n)$ for $d = 2$ (by Aggarwal and Suri [SoCG'87]) and near n^d for $d \geq 3$. We describe faster algorithms with running time

- $O(n2^{O(\log^* n)} \log n)$ for $d = 2$,
- $O(n^{2.5+o(1)})$ time for $d = 3$, and
- $\widetilde{O}(n^{(5d+2)/6})$ time for any constant $d \geq 4$.

To obtain the higher-dimensional result, we adapt and extend previous techniques for Klee's measure problem to optimize certain objective functions over the complement of a union of orthants.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Largest empty rectangle, largest empty box, Klee's measure problem

Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.24

Related Version *Full Version*: https://arxiv.org/abs/2103.08043

Funding Supported in part by NSF Grant CCF-1814026.

Acknowledgements I thank David Zheng for discussions on the 2D problem.

1 Introduction

Two dimensions. In the first part of this paper, we tackle the *largest empty rectangle* problem: Given a set P of n points in the plane and a fixed rectangle B_0, find the largest rectangle $B \subset B_0$ such that B does not contain any points of P in its interior. Here and throughout this paper, a "rectangle" refers to an axis-parallel rectangle; and unless stated otherwise, "largest" refers to maximizing the area.

The problem has been studied since the early years of computational geometry. While similar basic problems such as largest empty circle or largest empty square can be solved efficiently using Voronoi diagrams, the largest empty rectangle problem seems more challenging. The earliest reference on the 2D problem appears to be by Naamad, Lee, and Hsu in 1984 [26], who gave a quadratic-time algorithm. In 1986, Chazelle, Drysdale, and Lee [15] obtained an $O(n \log^3 n)$-time algorithm. Subsequently, at SoCG'87, Aggarwal and Suri [3] presented another algorithm requiring $O(n \log^3 n)$ time, followed by a more complicated second algorithm requiring $O(n \log^2 n)$ time. The $O(n \log^2 n)$ worst-case bound has not been improved since.[1]

A few results on related questions have been given. Dumitrescu and Jiang [20] examined the combinatorial problem of determining the worst-case number of maximum-area empty rectangles and proved an $O(n2^{\alpha(n)} \log n)$ upper bound; their proof does not appear to have

[1] Aggarwal and Suri's first algorithm can be sped up to run in near $O(n \log^2 n)$ time as well, since it relied on a subroutine for finding row minima in Monge staircase matrices, a problem for which improved results were later found [24, 12]; but these results do not appear to lower the cost of Aggarwal and Suri's second algorithm.

any implication to the algorithmic problem of finding a maximum-area empty rectangle. If the objective is changed to maximizing the perimeter, the problem is a little easier and an optimal $O(n \log n)$-time algorithm can already be found in Aggarwal and Suri's paper [3]. Another related problem of computing a maximum-area rectangle contained in a polygon has also been explored [16].

We obtain a new randomized algorithm that finds the maximum-area empty rectangle in $O(n2^{O(\log^* n)} \log n)$ expected time. This is not only an improvement of almost a full logarithmic factor over the previous 33-year-old bound, but is also close to optimal, except for the slow-growing iterated-logarithmic-like factor (as $\Omega(n \log n)$ is a lower bound in the algebraic decision tree model).

Our solution interestingly uses *interval trees* to efficiently divide the problem into subproblems of logarithmic size, yielding a recursion with $O(\log^* n)$ depth.

Higher dimensions. The higher-dimensional analog of the problem is *largest empty box*: Given a set P of n points in $\mathbb{R}^d$ and a fixed box B_0, find the largest box $B \subset B_0$ such that B does not contain any points of P in its interior. Here and throughout this paper, a "box" refers to an axis-parallel hyperrectangle; and unless stated otherwise, "largest" refers to maximizing the volume.

Several papers [18, 20, 19, 30] have studied related questions in higher dimensions, e.g., proving combinatorial bounds on the number of optimal boxes, or proving extremal bounds on the volume, or designing approximation algorithms. For the original (exact) computational problem, it is not difficult to obtain an algorithm that finds the largest empty box in $\widetilde{O}(n^d)$ time (for example, as was done by Backers and Keil [4]).[2] At the end of their SoCG'16 paper, Dumitrescu and Jiang [20] explicitly asked whether a faster algorithm is possible:

> "Can a maximum empty box in $\mathbb{R}^d$ for some fixed $d \geq 3$ be computed in $O(n^{\gamma_d})$ time for some constant $\gamma_d < d$?"

Dumitrescu and Jiang attempted to give a subcubic algorithm for the 3D problem, but their conditional solution required a sublinear-time dynamic data structure for finding the 2D maximum empty rectangles containing a query point – currently, the existence of such a data structure is not known.

On the lower bound side, Giannopoulos, Knauer, Wahlström, and Werner [23] proved that the largest empty box problem is $W[1]$-hard with respect to the dimension. This implies a conditional lower bound of $\Omega(n^{\beta d})$ for some absolute constant $\beta > 0$, assuming a popular conjecture on the hardness of the clique problem.

We answer the above question affirmatively. For $d = 3$, we give an $O(n^{5/2+\varepsilon})$-time algorithm, where $\varepsilon > 0$ is an arbitrarily small constant. For higher constant $d \geq 4$, we obtain an algorithm with an intriguing time bound that improves over n^d even more dramatically: $\widetilde{O}(n^{(5d+2)/6})$. For example, the bound is $O(n^{3.667})$ for $d = 4$, $\widetilde{O}(n^{4.5})$ for $d = 5$, and $O(n^{8.667})$ for $d = 10$.

Not too surprisingly, our 3D algorithm achieves subcubic complexity by applying standard range searching data structures (though the application is not be immediately obvious). Dynamic data structures are not used.

The techniques for our higher-dimensional algorithm are perhaps more original and significant, with potential impact to other problems. We first transform the largest empty box problem into a problem about a union of n orthants in $D = 2d$ dimensions (the

[2] Throughout the paper, $\widetilde{O}$ notation hides polylogarithmic factor.

transformation is simple and has been exploited before, such as in [5]). The union of orthants is known to have worst-case combinatorial complexity $O(n^{\lfloor D/2 \rfloor})$ [7]. Interestingly, we show that it is possible to maximize certain types of objective functions over the complement of the union, in time significantly smaller than the worst-case combinatorial complexity.

We accomplish this by adapting known techniques on Klee's measure problem [27, 10, 8, 11]. Specifically, we build on a remarkable method by Bringmann [8] for computing the volume of a union of n orthants in D dimensions in $O(n^{D/3+O(1)})$ time (the $O(1)$ term in the exponent was $2/3$ but has been later removed by author [11]). However, maximizing an objective function over the complement of the union is different from summing or integrating a function, and Bringmann's method does not immediately generalize to the former (for example, it exploits subtraction). We introduce extra ideas to extend the method, which results in a bigger time bound than $n^{D/3} = n^{2d/3}$ but nevertheless beats $n^{D/2} = n^d$. In particular, we use some simple graph-theoretical arguments, applied to graphs with $O(D)$ vertices.

Organization. We present our 2D algorithm in Sec. 2, our 3D algorithm in the full paper, and our higher-dimensional algorithms in Sec. 3–4 (all these parts may be read independently).

2 Largest empty rectangle in 2D

As in previous work [15, 3], we focus on solving a *line-restricted* version of the 2D largest empty rectangle problem: given a set P of n points below a fixed horizontal line ℓ_0 and a set Q of n points above ℓ_0, where the x-coordinates of all points have been pre-sorted, and given a rectangle B_0, find the largest-area rectangle $B \subset B_0$ that intersects ℓ_0 and is empty of points of $P \cup Q$. By standard divide-and-conquer, an $O(T(n))$-time algorithm for the line-restricted problem immediately yields an $O(T(n) \log n)$-time algorithm for the original largest empty rectangle problem, assuming that $T(n)/n$ is nondecreasing.

We begin by reformulating the line-restricted problem as a problem about horizontal line segments. In the subsequent subsections, we will work with this re-formulation.

For each point $p \in P$, let $s(p)$ be the longest horizontal line segment inside B_0 such that $s(p)$ passes through p and there are no points of P above $s(p)$. See Figure 1(a). We can compute $s(p)$ for all $p \in P$ in $O(n)$ time: this step is equivalent to the construction of the standard *Cartesan tree* [31, 22], for which there are simple linear-time algorithms (for example, by inserting points from left to right and maintaining a stack, like Graham's scan, as also re-described in previous papers [15, 3]). Similarly, for each $q \in Q$, let $t(q)$ be the longest horizontal line segment inside B_0 such that $t(q)$ passes through q and there are no points of Q below $t(q)$. We can also compute $t(q)$ for all $q \in Q$ in $O(n)$ time.

For a horizontal segment s, let x_s^- and x_s^+ denote the x-coordinates of its left and right endpoints respectively, and let y_s denote its y-coordinate. We say that a set S of horizontal segments is *laminar* if for every $s, s' \in S$, either the two intervals $[x_s^-, x_s^+]$ and $[x_{s'}^-, x_{s'}^+]$ are disjoint, or one interval is contained in the other (in other words, the intervals form a "balanceded parentheses" or tree structure). It is easy to see that for the segments defined above, $\{s(p) : p \in P\}$ is laminar and $\{t(q) : q \in Q\}$ is laminar.

The optimal rectangle must have some point $p^* \in P$ on its bottom side and some point $q^* \in Q$ on its top side (except when the optimal rectangle touches the bottom or top side of B_0, a case that can be easily dismissed in linear time). Chazelle, Drysdale, and Lee [15] already noted that the case when $[x_{s(p^*)}^-, x_{s(p^*)}^+]$ is contained in $[x_{t(q^*)}^-, x_{t(q^*)}^+]$ can be handled in $O(n)$ time (in their terminology, this is the case of "three supports in one half, one in the

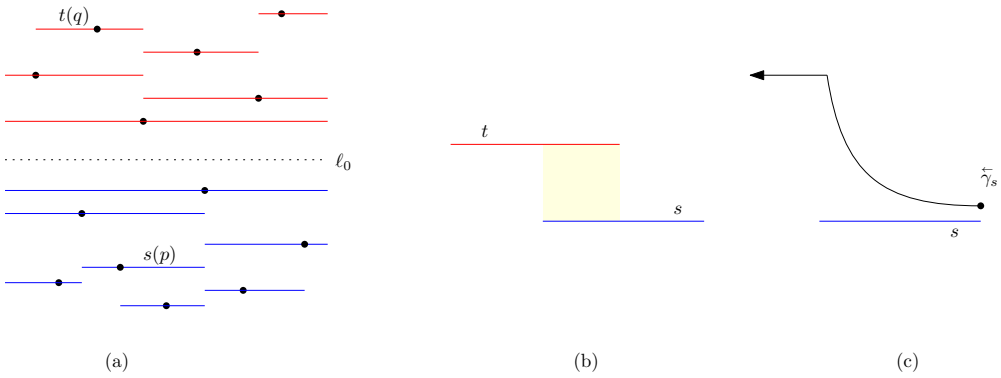


Figure 1 (a,b) Transforming points into horizontal segments. (c) Pseudo-ray $\overleftarrow{\gamma}_s$.

other").[3] The key remaining case is when $x^-_{t(q^*)} < x^-_{s(p^*)} < x^+_{t(q^*)} < x^+_{s(p^*)}$, where the area of the optimal rectangle is $(x^+_{t(q^*)} - x^-_{s(p^*)})(y_{t(q^*)} - y_{s(p^*)})$. All other cases are symmetric. The problem is thus reduced to the following (see Figure 1(b)):

▶ **Problem 1.** *Given a laminar set S of n horizontal segments and a laminar set T of n horizontal segments, where all x-coordinates have been pre-sorted, find a pair $(s,t) \in S \times T$ such that $x^-_t < x^-_s < x^+_t < x^+_s$, maximizing $(x^+_t - x^-_s)(y_t - y_s)$.*

We find it more convenient to work with the corresponding *decision problem*, as stated below. By the author's randomized optimization technique [9], an $O(T(n))$-time algorithm for Problem 2 yields an $O(T(n))$-expected-time algorithm for Problem 1, assuming that $T(n)/n$ is nondecreasing:

▶ **Problem 2.** *Given a laminar set S of n horizontal segments and a laminar set T of n horizontal segments, where all x-coordinates have been pre-sorted, and given a value $r > 0$, decide if there exists a pair $(s,t) \in S \times T$ such that $x^-_t < x^-_s < x^+_t < x^+_s$ and $(x^+_t - x^-_s)(y_t - y_s) > r$, and if so, report one such pair. We call such a pair* good.

2.1 Preliminaries

To help solve Problem 2, we define a curve γ_s for each $s \in S$:

$$\gamma_s(x) = \begin{cases} \frac{r}{x - x^-_s} + y_s & \text{if } x \geq x^-_s + \delta \\ M + x^-_s & \text{if } x < x^-_s + \delta, \end{cases}$$

for a sufficiently small $\delta > 0$ and a sufficiently large $M = M(\delta)$. (The main first part of the curve is a hyperbola.) The condition $(x^+_t - x^-_s)(y_t - y_s) > r$ is met iff the point (x^+_t, y_t) (i.e., the right endpoint of t) is above the curve γ_s, assuming $x^+_t \geq x^-_s + \delta$. Note that these curves form a family of *pseudo-lines*, i.e., every pair of curves intersect at most once: this can be seen from the fact that for any two curves γ_s and $\gamma_{s'}$ with $x^-_s \geq x^-_{s'}$, the difference $\gamma_s(x) - \gamma_{s'}(x) = \frac{r(x^-_s - x^-_{s'})}{(x - x^-_s)(x - x^-_{s'})} + y_s - y_{s'}$ is nonincreasing for $x \geq x^-_s$.

[3] The solution is simple: for each $p \in P$, we find the lowest point $q_p \in Q$ with x-coordinate in the interval $[x^-_{s(p)}, x^+_{s(p)}]$, and take the maximum of $(x^+_{s(p)} - x^-_{s(p)})(y_{t(q_p)} - y_{s(p)})$. All these lowest points q_p can be found "bottom-up" in the tree formed by the intervals $\{[x^-_{s(p)}, x^+_{s(p)}] : p \in P\}$, in linear total time.

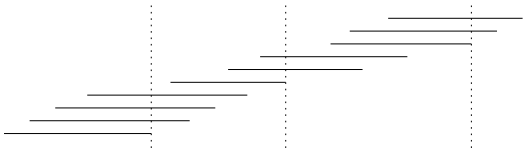


Figure 2 Proof of Lemma 1(b): the x-projected intervals and the division into slabs.

Define the curve segment $\overleftarrow{\gamma}_s$ to be the part of γ_s restricted to $x \leq x_s^+$. (See Figure 1(c).) These curve segments form a family of *pseudo-rays*. The lower envelope of n pseudo-rays has at most $2n$ edges, by known combinatorial bounds on order-2 Davenport-Schinzel sequences [29]. The following lemma summarizes known subroutines we need on the computation of lower envelopes (proofs are briefly sketched).

▶ **Lemma 1.** *Consider a set of n pseudo-lines, sorted by their* pseudo-slopes*, such that if γ and γ' intersects and γ has smaller pseudo-slope, then γ is above γ' to the left of the intersection. Assume that the intersection of any two pseudo-lines can be computed in constant time.*

(a) *Consider n pseudo-rays that are parts of the given pseudo-lines, such that the x-coordinates of the left endpoints are all $-\infty$, and the x-coordinates of the right endpoints are monotone (increasing or decreasing) in the pseudo-slopes. Then the lower envelope of these pseudo-rays can be computed in $O(n)$ time.*

(b) *Consider n pseudo-segments that are parts of the given pseudo-lines, such that x-coordinates of the left endpoints are monotone in the pseudo-slopes and the x-coordinates of the right endpoints are monotone in the pseudo-slopes. Then the lower envelope of these pseudo-segments can be computed in $O(n)$ time.*

Proof. Part (a) follows by a straightfoward variant of Graham's scan [17] (originally for computing planar convex hulls, or by duality, lower envelopes of lines). We insert pseudo-rays in decreasing order of their right endpoints' x-values, while maintaining the portion of the lower envelope to the left of the right endpoint of the current pseudo-ray. In each iteration, by the monotonicity assumption, a prefix or suffix of the lower envelope gets deleted (i.e., popped from a stack).

For part (b), the main case is when both the left and right endpoints are monotonically increasing in the pseudo-slopes (the case when both are monotonically decreasing is symmetric, and the case when they are monotone in different directions easily reduces to two instances of the pseudo-ray case). Greedily construct a minimal set of vertical lines that stab all the pseudo-segments: namely, draw a vertical line at the leftmost right endpoint, remove all pseudo-segments stabbed, and repeat. This process can be done in $O(n)$ time by a linear scan. These vertical lines divide the plane into slabs. (See Figure 2.) In each slab, the pseudo-segments behave like pseudo-rays, so we can compute the lower envelope inside the slab in linear time by applying part (a) twice, for the leftward rays and for the rightward rays (the two envelopes can be merged in linear time). Since each pseudo-segment participates in at most two slabs, the total time is linear. ◀

As an application of Lemma 1(b), we mention an efficient algorithm for a special case of Problem 2, which will be useful later.

▶ **Corollary 2.** *In the case when all segments in S and T intersect a fixed vertical line, Problem 2 can be solved in $O(n)$ time.*

Proof. Since S and T are laminar, the x-projected intervals in each set are nested. Let $s_1, s_2, \ldots$ be the segments in S with $[x^-_{s_1}, x^+_{s_1}] \subseteq [x^-_{s_2}, x^+_{s_2}] \subseteq \cdots$, and let $t_1, t_2, \ldots$ be the segments in T with $[x^-_{t_1}, x^+_{t_1}] \subseteq [x^-_{t_2}, x^+_{t_2}] \subseteq \cdots$. For each s_i, let $a(i)$ be the smallest index with $x^-_{t_{a(i)}} < x^-_{s_i}$, let $b(i)$ be the smallest index with $x^-_{s_i} < x^+_{t_{b(i)}}$, and let $c(i)$ be the largest index with $x^+_{t_{c(i)}} < x^+_{s_i}$. Note that $a(i)$ is monotonically increasing in i, and $b(i)$ is monotonically decreasing in i, and $c(i)$ is monotonically increasing in i. It is straightforward to compute $a(i), b(i), c(i)$ for all i by a linear scan.

The problem reduces to finding a pair (s_i, t_j) such that $\max\{a(i), b(i)\} \leq j \leq c(i)$ and the right endpoint of t_j is above γ_{s_i}. Define the curve segment $\overline{\gamma}_{s_i}$ to be the part of γ_{s_i} restricted to $x \in [\max\{x^+_{t_{a(i)}}, x^+_{t_{b(i)}}\}, x^+_{t_{c(i)}}]$. The problem reduces to finding a t_j whose right endpoint is above some curve segment $\overline{\gamma}_{s_i}$, i.e., above the lower envelope of these curve segments. We can compute this lower envelope in $O(n)$ time by Lemma 1(b) (more precisely, by two invocations of the lemma, as $\max\{x^+_{t_{a(i)}}, x^+_{t_{b(i)}}\}$ consists of a monotonically increasing and a monotonically decreasing part). The problem can be then be solved by linear scan over the envelope and the endpoints of t_j. ◀

2.2 Algorithm

We are now ready to describe our new algorithm for solving Problem 2, using interval trees and an interesting recursion with $O(\log^* n)$ depth.

▶ **Theorem 3.** *Problem 2 can be solved in $O(n2^{O(\log^* n)})$ time.*

Proof. As a first step, we build the standard *interval tree* for the given horizontal segments in $S \cup T$. This is a perfectly balanced binary tree of with $O(\log n)$ levels, where each node corresponds to a vertical slab. The root slab is the entire plane, the slab at a node is the union of the slabs of its two children, and each leaf slab contains no endpoints in its interior. Each segment is stored in the lowest node v whose slab contains the segment (i.e., the segment is contained in v's slab but is not contained in either child's subslab). Note that each segment is stored only once (unlike in another standard structure called the "segment tree"). We can determine the slab containing each segment in $O(1)$ time by an LCA query [6] (which is easier in the case of a perfectly balanced binary tree).

For each node v, let S_v (resp. T_v) be the set of all segments of S (resp. T) stored in v. Define the *level* of a segment to be the level of the node it is stored in.

Case 1. There exists a good pair (s^*, t^*) where s^* and t^* have the same level. Here, s^* and t^* must be stored in the same node v of the interval tree. Thus, a good pair can be found as follows:

1. For each node v, solve the problem for S_v and T_v by Corollary 2 in $O(|S_v| + |T_v|)$ time. Note that all segments in $S_v \cup T_v$ indeed intersect a fixed vertical line (the dividing line at v).

The total running time of this step is $O(n)$, since each segment is in only one S_v or T_v.

Case 2. There exists a good pair (s^*, t^*) where s^* is on a strictly lower level than t^*. To deal with this case, we perform the following steps, for some choice of parameter $b \geq \log n$:

2a. For each node v, compute the lower envelope of the pseudo-rays $\{\overleftarrow{\gamma}_s : s \in S_v\}$ by Lemma 1(a) in $O(|S_v|)$ time; let $\mathcal{E}_v$ denote this envelope restricted to v's slab. Note that because all segments in S_v intersect a fixed vertical line and S_v is laminar, the x^+_s values are monotonically decreasing in the x^-_s values for $s \in S_v$ and so are indeed monotone in the pseudo-slopes of these pseudo-rays.

2b. Divide the plane into a set Σ of n/b vertical slabs each containing b right endpoints of T.
2c. For each slab $\sigma \in \Sigma$,
- let T_σ be the set of all segments $t \in T$ with right endpoints in σ, and
- let S_σ be the set of all segments $s \in S$ such that $\overleftarrow{\gamma}_s$ appears on $\mathcal{E}_v \cap \sigma$ for some node v.

Divide S_σ (arbitrarily) into blocks of size b and recursively solve the problem for T_σ and each block of S_σ.

Correctness. Consider a good pair (s^*, t^*) with s^* on a strictly lower level than t^*. Let σ be the slab in Σ containing the right endpoint of t^*, i.e., $t^* \in T_\sigma$. Let v be the node s^* is stored in. Then t^* intersects the left wall of the slab at v (since t^* must be stored in a proper ancestor of v). Now, the right endpoint of t^* is below $\overleftarrow{\gamma}_{s^*}$ and is thus below $\mathcal{E}_v$. Let $\overleftarrow{\gamma}_s$ be the curve on $\mathcal{E}_v$ that the right endpoint of t^* is below, with $s \in S_v$. Then $\overleftarrow{\gamma}_s$ appears on $\mathcal{E}_v \cap \sigma$, and so $s \in S_\sigma$. Since the right endpoint of t^* is below $\overleftarrow{\gamma}_s$, we have $x_s^- < x_{t^*}^+ < x_s^+$, and since t^* intersects the left wall of v's slab, we have $x_{t^*}^- < x_s^-$. So, (s, t^*) is good, and the recursive call for T_σ and some block of S_σ will find a good pair.

Analysis. The total number of edges in all envelopes $\mathcal{E}_v$ is at most $2\sum_v |S_v| \le 2n$. Since the envelopes $\mathcal{E}_v$ have disjoint x-projections for nodes v at the same level, and since there are $O(\log n)$ levels, the $O(n/b)$ dividing vertical lines of Σ intersect at most $O((n/b)\log n)$ edges among all the envelopes. Thus, $\sum_{\sigma\in\Sigma} |S_\sigma| \le 2n + O((n/b)\log n) = O(n)$ if $b \ge \log n$, and so the total number of recursive calls in step 2c is $O(n/b)$.

Case 3. There exists a good pair (s^*, t^*) where s^* is on a strictly higher level than t^*. This remaining case is symmetric to Case 2 (by switching S and T and negating y-coordinates).

Total time. By running the algorithms for all three cases, a good pair is guaranteed to be found if one exists. The running time satisfies the recurrence $T(n) \le O(n/b)T(b) + O(n)$. Setting $b = \log n$ gives $T(n) \le n2^{O(\log^* n)}$. ◀

By the observations from the beginning of this section, we can now solve Problem 1 and the line-restricted problem in $O(n2^{O(\log^* n)})$ expected time, and the original largest empty rectangle problem in $O(n2^{O(\log^* n)} \log n)$ expected time.

▶ **Corollary 4.** *Given n points in $\mathbb{R}^2$ and a rectangle B_0, we can compute the maximum-area empty rectangle inside B_0 in $O(n2^{O(\log^* n)} \log n)$ expected time.*

3 Largest empty anchored box in higher dimensions (warm-up)

To prepare for our solution to the largest empty box problem in higher constant dimensions, we first investigate a simpler variant, the *largest empty anchored box* problem: given a set P of n points in $\mathbb{R}^d$ and a fixed box B_0, find the largest-volume anchored box in B_0 that does not contain any points of P in its interior, where an *anchored* box has the form $B = (0, x_1) \times \cdots (0, x_d)$ (having the origin as one of its vertices).

Let $\bigcup S$ denote the union of a set S of objects. By mapping a box $B = (0, x_1) \times \cdots (0, x_d)$ to the point $(x_1, \ldots, x_d)$, and mapping each input point $(p_1, \ldots, p_d)$ to the orthant $(p_1, \infty) \times \cdots \times (p_d, \infty)$, the largest empty anchored box problem reduces to:

▶ **Problem 3.** *Define the function $H_{\text{vol}}(x_1, \ldots, x_d) = x_1 x_2 \cdots x_d$. Given a set S of n orthants in $\mathbb{R}^d$ and a box B_0, find the maximum of H_{vol} over $B_0 - \bigcup S$.*

By known results [7], the union of n orthants in $\mathbb{R}^d$ has worst-case combinatorial complexity $O(n^{\lfloor d/2 \rfloor})$ and can be constructed in $\widetilde{O}(n^{\lfloor d/2 \rfloor})$ time. We will show that Problem 3 can be solved faster than explicitly constructing the union.

3.1 Preliminaries

A key tool we need is a spatial partitioning scheme due to Overmars and Yap [27] (originally developed for solving Klee's measure problem in $\widetilde{O}(n^{d/2})$ time). The version stated below is taken from [11, Lemma 4.6]; see that paper for a short proof. (The partitioning scheme is also related to "orthogonal BSP trees" [21, 14].)

▶ **Lemma 5.** *Given a set of n axis-parallel flats (of possibly different dimensions) in $\mathbb{R}^d$, and given a parameter r, we can divide $\mathbb{R}^d$ into $O(r^d)$ cells (bounded and unbounded boxes) so that each cell intersects $O(n/r^j)$ $(d-j)$-flats.*

The construction of the cells, along with the conflict lists (lists of all flats intersecting each cell), can be done in $\widetilde{O}(n + r^d + K)$ time,[4] *where K is the total size of the conflict lists.*

Call a function $H : \mathbb{R}^d \to \mathbb{R}$ *simple* if it has the form

$$H(x_1, \ldots, x_d) \;=\; h_1(x_1) \cdots h_d(x_d),$$

where each h_i is a univariate step function. The *complexity* of H refers to the total complexity (number of steps) in these step functions. As an illustration of the usefulness of Lemma 5, we first how to maximize simple functions over the complement of a union of orthants in $\widetilde{O}(n^{d/2})$ time:

▶ **Lemma 6.** *Let H be a simple function with $O(n)$ complexity. Given a set S of n orthants in $\mathbb{R}^d$ and a box B_0, we can compute the maximum of H in $B_0 - \bigcup S$ in $\widetilde{O}(n^{d/2})$ time for any constant $d \geq 2$.*

Proof. Apply Lemma 5 to the $O(n)$ $(d-2)$-flats that pass through the $(d-2)$-faces of the given orthants. This yields a partition of B_0 into cells.

Consider a cell Δ. The number of $(d-2)$-flats intersecting Δ is bounded by $O(n/r^2)$, which can be made 0 by setting $r := \Theta(\sqrt{n})$. Consequently, only $(d-1)$-faces of the given orthants may intersect Δ, i.e., all orthants are 1-sided inside Δ. The union of 1-sided orthants simplifies to the complement of a box (we can use orthogonal range searching or intersection data structure to identify the 1-sided orthants intersecting Δ and compute this box in $\widetilde{O}(1)$ time [1, 17]). For a simple function $H(x_1, \ldots, x_d) = h_1(x_1) \cdots h_d(x_d)$, we can maximize H over a box by maximizing $h_i(x_i)$ over an interval for each $i \in \{1, \ldots, d\}$ separately. This corresponds to a 1D range maximum query for each i, which can be done straightforwardly in $O(\log n)$ time (or more carefully in $O(1)$ time [6]). As the number of cells is $O(r^d) = O(n^{d/2})$, the total running time is $\widetilde{O}(n^{d/2})$. ◀

3.2 Algorithm

To improve over $n^{d/2}$, we adapt an approach by Bringmann [8] (originally for solving Klee's measure problem for orthants in $O(n^{d/3+O(1)})$ time). The approach involves first solving the 2-sided special case, and then applying Overmars and Yap's partitioning scheme. A *2-sided orthant* is the set of all points $(x_1, \ldots, x_d) \in \mathbb{R}^d$ satisfying a condition of the form $[x_i \;?\; a] \wedge [x_j \;?\; b]$ for some $i, j \in \{1, \ldots, d\}$, where each occurrence of "?" is either $\leq$ or $\geq$. We will adapt the author's subsequent re-interpretation [11, Section 4.1] of Bringman's technique, described in terms of monotone step functions.

▶ **Theorem 7.** *In the case when all the input orthants are 2-sided, Problem 3 can be solved in $\widetilde{O}(n^{\lfloor d/2 \rfloor/2})$ time for any constant $d \geq 4$.*

[4] A weaker time bound was stated in [11, Lemma 4.6], but the output-sensitive time bound follows directly from the same construction.

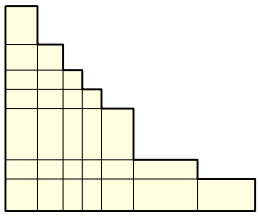


Figure 3 The union of one type of 2-sided orthants.

Proof. The boundary of the union of 2-sided orthants of the form $[x_i \ ? \ a] \wedge [x_j \ ? \ b]$ with a fixed i, j and a fixed choice for the two "?"s is a staircase, i.e., the graph of a univariate monotone (increasing or decreasing) step function. (See Figure 3.) Thus, the complement of the union of 2-sided orthants can be expressed as the set of all points $(x_1, \ldots, x_d) \in \mathbb{R}^d$ satisfying an expression $E(x_1, \ldots, x_d)$ which is a conjunction of $O(d^2)$ predicates each of the form $[x_i \ ? \ f(x_j)]$, where $i, j \in \{1, \ldots, d\}$, "?" is $\leq$ or $\geq$, and f is a monotone step function. The total complexity of these step functions is $O(n)$. Conversely, any such expression can be mapped back to the complement of a union of $O(n)$ 2-sided orthants.

We first observe a few simple rules for rewriting expressions:

1. $[x_i \leq f(x_j)] \wedge [x_i \leq g(x_j)]$ can be rewritten as $[x_i \leq \min\{f, g\}(x_j)]$ if f and g are both increasing or both decreasing. Note that the lower envelope $\min\{f, g\}$ is still a monotone step function with $O(n)$ complexity. A similar rule applies for $\geq$.
2. $[x_i \leq f(x_j)]$ can be rewritten as $[x_j \geq f^{-1}(x_i)]$ if f is increasing (the inequality is flipped if f is decreasing). Note that the inverse f^{-1} is still a monotone step function.
3. More generally, $[f(x_i) \leq g(x_j)]$ can be rewritten as $[x_j \geq (f^{-1} \circ g)(x_i)]$ if f is increasing (the inequality is flipped if f is decreasing). Note that the composition $f^{-1} \circ g$ is still a monotone step function with $O(n)$ complexity.
4. $[x_i \leq f(x_j)] \wedge [x_i \leq g(x_k)]$ can be rewritten as the disjunction of $[x_i \leq f(x_j)] \wedge [f(x_j) \leq g(x_k)]$ and $[x_i \leq g(x_k)] \wedge [g(x_k) \leq f(x_j)]$. A similar rule applies for $\geq$.

The plan is to decrease the dimension by repeatedly eliminating variables:

We maintain a simple function H. Initially, $H(x_1, \ldots, x_d) = \sigma(x_1) \cdots \sigma(x_d)$, where $\sigma(x)$ denotes the successor of x among the $O(n)$ input coordinate values (σ is a step function). We call an index i *free* if the variable x_i appears exactly once in H and is "unaltered", i.e., $h_i(x_i) = \sigma(x_i)$. All indices are initially free.

In each iteration, we pick a free index i. Whenever x_i appears more than twice in E, we can apply rule 4 (in combination with rules 1–3) to obtain a disjunction of 2 subexpressions, where in each subexpression, the number of occurrences of x_i is decreased. By repeating this process $O(1)$ times (recall that d is a constant), we obtain a disjunction of $O(1)$ subexpressions, where in each subexpression, only at most two occurrences of x_i remain – in at most one predicate of the form $[x_i \leq f(x_j)]$, and at most one predicate of the form $[x_i \geq g(x_k)]$.

We branch off to maximize H over each of these subexpressions separately. In such a subexpression, to eliminate the variable x_i while maximizing H, we replace the two predicates $[x_i \leq f(x_j)]$ and $[x_i \geq g(x_k)]$ with $[f(x_j) \geq g(x_k)]$, and replace x_i with $f(x_j)$ in H (i.e., reset $h_j(x_j)$ to $h_j(x_j)\sigma(f(x_j))$, which is still a step function with $O(n)$ complexity). Now, i and j are not free.

We stop a branch when there are no free indices left. At the end, we get a large but $O(1)$ number of subproblems, where in each subproblem, at least $\lceil d/2 \rceil$ variables have been eliminated, i.e., the dimension is decreased to $d' \leq \lfloor d/2 \rfloor$. We solve each subproblem by Lemma 6 in $\widetilde{O}(n^{d'/2})$ time. ◀

We now combine Theorem 7 and Lemma 5 to solve Problem 3:

▶ **Corollary 8.** *Problem 3 can be solved in $\widetilde{O}(n^{d/3+\lfloor d/2\rfloor/6})$ time for any constant $d \geq 4$.*

Proof. Apply Lemma 5 to the $O(n)$ $(d-3)$-flats and $(d-2)$-flats through the $(d-3)$-faces and $(d-2)$-faces of the given orthants. This yields a partition of B_0 into cells.

Consider a cell Δ. The number of $(d-3)$-flats intersecting Δ is $O(n/r^3)$, which can be made 0 by setting $r := \Theta(n^{1/3})$. The number of $(d-2)$-flats intersecting Δ is $O(n/r^2) = O(n^{1/3})$. So, inside the cell Δ, all orthants are 2-sided or 1-sided, with $O(n^{1/3})$ 2-sided orthants. The union of 1-sided orthants simplifies to the complement of a box (we can use orthogonal range searching or intersection data structures [1, 17] to identify the 1-sided orthants intersecting Δ and compute this box). We can thus apply Theorem 7 to maximize H over the cell Δ in $\widetilde{O}((n^{1/3})^{\lfloor d/2\rfloor/2})$ time. As there are $O(r^d) = O(n^{d/3})$ cells, the total running time is $\widetilde{O}(n^{d/3} \cdot (n^{1/3})^{\lfloor d/2\rfloor/2})$. ◀

▶ **Corollary 9.** *Given n points in $\mathbb{R}^d$ and a box B_0, we can compute the maximum-volume empty anchored box inside B_0 in $\widetilde{O}(n^{d/3+\lfloor d/2\rfloor/6}) \leq \widetilde{O}(n^{5d/12})$ time for any constant $d \geq 4$.*

4 Largest empty box in higher dimensions

We now adapt the approach from Section 3 to solve the original largest empty box problem in higher constant dimensions. By d levels of divide-and-conquer, it suffices to solve the *point-restricted* version of the problem: given a set P of n points in $\mathbb{R}^d$, a fixed box B_0, and a fixed point o, find the largest-volume box $B \subset B_0$ that contains o and is empty of points of P. An $O(T(n))$-time algorithm for the point-restricted problem immediately yields an $O(T(n)\log^d n)$-time algorithm for the original problem (in fact, the polylogarithmic factor disappears if $T(n)/n^{1+\delta}$ is increasing for some constant $\delta > 0$). Without loss of generality, assume that o is the origin.

By mapping a box $B = (-x_1, x_1') \times \cdots \times (-x_d, x_d')$ (which has volume $(x_1 + x_1') \cdots (x_d + x_d')$) to the point $(x_1, x_1', \ldots, x_d, x_d')$ in $2d$ dimensions, and mapping each input point $p = (p_1, \ldots, p_d)$ to the orthant $(-p_1, \infty) \times (p_1, \infty) \times \cdots (-p_d, \infty) \times (p_d, \infty)$ in $2d$ dimensions (and changing B_0 appropriately), the problem reduces to the following variant of Problem 3, after doubling the dimension:

▶ **Problem 4.** *Define the function $H_{\text{new-vol}}(x_1, \ldots, x_d) = (x_1 + x_2)(x_3 + x_4) \cdots (x_{d-1} + x_d)$ for an even d. Given a set S of n orthants in $\mathbb{R}^d$ and a box B_0, find the maximum of $H_{\text{new-vol}}$ over $B_0 - \bigcup S$.*

The above objective function $H_{\text{new-vol}}$ is a bit more complicated than the one from Section 3, and so further ideas are needed...

4.1 Preliminaries

For a multigraph G with vertex set $\{1, \ldots, d\}$ (without self-loops), define a *G-function* $H : \mathbb{R}^d \to \mathbb{R}$ to be a function of the form

$$H(x_1, \ldots, x_d) \;=\; \prod_{i=1}^{d} h_i(x_i) \cdot \prod_{e=ij \in G} (h_e'(x_i) + h_e''(x_j)),$$

where h_i, h_e', and h_e'' are univariate step functions. The *complexity* of H refers to the total complexity of these step functions.

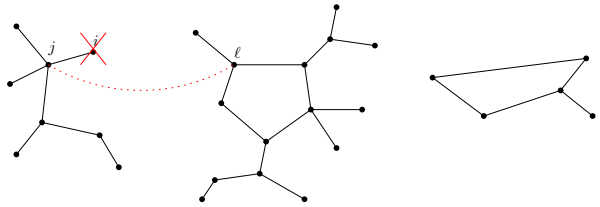


Figure 4 After removing vertex i and adding edge $j\ell$, the graph G remains a pseudo-forest.

A *pseudo-forest* is a graph where each component is either a tree, or a tree plus an edge – in the latter case, the component is called a *1-tree* (and we allow the extra edge to be a duplicate of an edge in the tree).

▶ **Lemma 10.** *Let H be a G-function H with $O(n)$ complexity. Given a box B_0, we can compute the maximum of H over B_0 in $\widetilde{O}(n)$ time if G is a forest, or $\widetilde{O}(n^2)$ time if G is a pseudo-forest, for any constant d.*

Proof. For the forest case: Pick a leaf i. Then H is of the form $h(x_i) \cdot (h'(x_i) + h''(x_j)) \cdots$, where h, h', h'' are step functions and x_i does not appear in "$\cdots$". Define $F(\xi) := \max_{x \in \mathbb{R}} h(x) \cdot (h'(x) + \xi)$. Then F is the upper envelope of $O(n)$ linear functions in the single variable ξ, and can be constructed in $\widetilde{O}(n)$ time by the dual of a planar convex hull algorithm [17]. We can eliminate the variable x_i by replacing the $h(x_i) \cdot (h'(x_i) + h''(x_j))$ factor with $F(h''(x_j))$ (which is a step function in x_j with $O(n)$ complexity). As a result, H becomes a $(G - \{i\})$-function in $d - 1$ variables. After d iterations, the problem becomes trivial.

For the pseudo-forest case: We may assume the graph is connected, since we can maximize the parts of H corresponding to different components separately. Pick a vertex i that belongs to the unique cycle of G (if exists). Then $G - \{i\}$ is a forest. By trying out all $O(n)$ different settings of x_i (breakpoints of the step functions), the problem reduces to $O(n)$ instances of the forest case. ◀

▶ **Lemma 11.** *Let H be a G-function with $O(n)$ complexity, where G is a pseudo-forest. Given a set S of n boxes in $\mathbb{R}^d$ and a box B_0, we can compute the maximum of H over $B_0 - \bigcup S$ in $\widetilde{O}(n^{d/2+1})$ time for any constant d.*

Proof. Apply Lemma 5 to the $O(n)$ $(d-2)$-flats through the boundaries of the orthants, together with the $O(n)$ $(d-1)$-flats $x_j = a$ for all breakpoints a of the step functions appearing in H. This yields a partition of B_0 into cells.

Consider a cell Δ. The number of $(d-2)$-flats intersecting Δ is $O(n/r^2)$, which can be made 0 by setting $r := \Theta(\sqrt{n})$. So, inside the cell Δ, we see only 1-sided orthants, and their union simplifies to the complement of a box. In addition, the number of $(d-1)$-flats intersecting Δ is $O(n/r) = O(\sqrt{n})$; in other words, the breakpoints of the step functions in H relevant to the cell Δ is $O(\sqrt{n})$. We can thus apply Lemma 10 to maximize H over the cell Δ in $\widetilde{O}((\sqrt{n})^2)$ time. As the number of cells is $O(r^d) = O(n^{d/2})$, the total running time is $\widetilde{O}(n^{d/2} \cdot (\sqrt{n})^2)$. ◀

4.2 Algorithm

We now modify the proof of Theorem 7 to solve Problem 4 for the 2-sided orthant case:

▶ **Theorem 12.** *In the case when all input orthants are 2-sided, Problem 4 can be solved in $\widetilde{O}(n^{d/4+1})$ time for any constant even d.*

Proof. We maintain a G-function H. Initially, $H(x_1, \ldots, x_d) = (\sigma(x_1) + \sigma(x_2))(\sigma(x_3) + \sigma(x_4)) \cdots (\sigma(x_{d-1}) + \sigma(x_d))$, with G being a matching with $d/2$ edges, where $\sigma(x)$ denotes the successor of x among all $O(n)$ input coordinate values. We call an index i *free* if x_i appears exactly once in H and is "unaltered" (i.e., H is of the form $(\sigma(x_i) + h(x_\ell)) \cdots$ where x_i does not appear in "$\cdots$"). All indices are initially free. We maintain the following invariants: at any time, (i) G is a pseudo-forest with at most $d/2$ edges, and (ii) for each component T of G which is a tree (not a 1-tree), T has at least two free leaves.

In each iteration, we pick a free leaf i in some component T of G which is a tree. As before, we rewrite the expression E as a disjunction of $O(1)$ subexpressions, where in each subexpression, only two occurrences of x_i remain – in a predicate of the form $[x_i \le f(x_j)]$, and another predicate of the form $[x_i \ge g(x_k)]$.

We branch off to maximize H for each of these subexpressions separately. In such a subexpression, to eliminate the variable x_i while maximizing H, we replace the two predicates $[x_i \le f(x_j)]$ and $[x_i \ge g(x_k)]$ with $[f(x_j) \ge g(x_k)]$, and replace x_i with $f(x_j)$ in H (since x_i is free). Now, i and j are not free. Also, in the graph G, the unique edge $i\ell$ incident to i is replaced by $j\ell$ (unless $j = \ell$). If j is in the same component T as i, then T becomes a 1-tree; otherwise, two components are merged and the new component is either a tree with at least two free leaves, or a 1-tree. (See Figure 4.) So, the invariants are maintained.

We stop a branch when there are no free indices left. At the end, we get $O(1)$ subproblems, where in each subproblem, all components are 1-trees, and so the number of nodes is exactly equal to the number of edges, implying that the dimension is $d' \le d/2$. Now we can apply Lemma 11 to solve these subproblems in $\widetilde{O}(n^{d'/2+1})$ time. ◀

▶ **Corollary 13.** *Problem 4 can be solved in $\widetilde{O}(n^{(5d+4)/12})$ time for any constant even d.*

Proof. Following the proof of Corollary 8 but using Theorem 12 instead of Theorem 7 gives running time $\widetilde{O}(n^{d/3} \cdot (n^{1/3})^{d/4+1})$. ◀

Applying the above corollary in $2d$ dimensions, we finally obtain:

▶ **Corollary 14.** *Given n points in $\mathbb{R}^d$ and a box B_0, we can compute the maximum-volume empty box inside B_0 in $\widetilde{O}(n^{(5d+2)/6})$ time for any constant d.*

5 Remarks

On the 2D algorithm. The $2^{O(\log^* n)}$ factor can be analyzed more precisely (an upper bound of $3^{\log^* n}$ can be shown with minor changes to the algorithm). A question remains whether the extra factor could be further lowered to inverse-Ackermann, or eliminated completely.

The previous algorithm by Aggarwal and Suri [3] used matrix searching techniques, namely, for finding row minima in certain types of partial Monge matrices. We are able to bypass such subroutines because we have focused our effort on solving the *decision problem* (due to the author's randomized optimization technique [9]). Generally, the row minima problem is equivalent to the computation of lower envelopes of pseudo-rays and pseudo-segments, not necessarily of constant complexity [12]. However, to solve the decision problem, we only need lower envelopes of pseudo-rays and pseudo-segments of constant complexity (formed by hyperbolas), for which there are simpler direct methods, as we have noted in Lemma 1. (Incidentally, the proof we gave for reducing Lemma 1(b) to (a) is essentially equivalent to Aggarwal and Klawe's reduction of row minima in double-staircase to staircase matrices [2]; a similar idea has also been used in dynamic data structures with "FIFO updates" [13].)

On the other hand, it should be possible to modify our approach to get improved *deterministic* algorithms for 2D largest empty rectangle, by solving the optimization problem directly and using known matrix searching subroutines [24], though details are more involved and the running time seems slightly worse than in our randomized algorithm.

It is theoretically possible to devise an optimal algorithm for Problem 1 without knowing the true complexity of the algorithm, since by a constant number of rounds of recursion in our method, the problem is reduced to subproblems of very small size (say, $\log\log\log\log n$), for which we can afford to explicitly build an optimal decision tree (this type of trick appeared before in the literature [25, 28]).

On the higher-dimensional algorithms. Our approach in higher dimensions works for maximizing the perimeter (sum of edge lengths) of the box as well. In fact, the algorithm for the simpler, largest empty *anchored* box problem should suffice here after doubling the dimension, since the required objective function here is $H_{\text{perim}}(x_1, \ldots, x_d) = x_1 + \cdots + x_d$, which is "similar" to $H_{\text{vol}}(x_1, \ldots, x_d) = x_1 \cdots x_d$.

For the largest empty anchored box problem, the $\widetilde{O}(n^{5d/12})$ time bound can be further improved to $\widetilde{O}(n^{(7d+6)/18})$, by building on the graph-theoretic ideas from Section 4, as we show in the full paper. Still further improvements of the exponent is likely possible, by working with G-functions for *hypergraphs* G, not just graphs, though improvement on the fraction 7/18 appears very tiny and requires d to be a very large constant, and the algorithm becomes more complicated. For the largest empty box problem, we currently don't know how to improve the fraction 5/6, even using hypergraphs. It remains a fascinating question what the best fraction β is for which the problem could be solved in $O(n^{\beta d+o(d)})$ time.

On the conditional lower bound side, another relevant question is whether Problem 3 or 4 remain $W[1]$-hard with respect to the parameter d in the special case of 2-sided orthants.

References

1. Pankaj K. Agarwal and Jeff Erickson. Geometric range searching and its relatives. In B. Chazelle, J. E. Goodman, and R. Pollack, editors, *Advances in Discrete and Computational Geometry*, pages 1–56. AMS Press, 1999. URL: `http://jeffe.cs.illinois.edu/pubs/survey.html`.
2. Alok Aggarwal and Maria M. Klawe. Applications of generalized matrix searching to geometric algorithms. *Discret. Appl. Math.*, 27(1-2):3–23, 1990. `doi:10.1016/0166-218X(90)90124-U`.
3. Alok Aggarwal and Subhash Suri. Fast algorithms for computing the largest empty rectangle. In *Proc. 3rd Symposium on Computational Geometry (SoCG)*, pages 278–290, 1987. `doi:10.1145/41958.41988`.
4. Jonathan Backer and J. Mark Keil. The mono- and bichromatic empty rectangle and square problems in all dimensions. In *Proc. 9th Latin American Theoretical Informatics Symposium (LATIN)*, volume 6034 of *Lecture Notes in Computer Science*, pages 14–25. Springer, 2010. `doi:10.1007/978-3-642-12200-2_3`.
5. Jérémy Barbay, Timothy M. Chan, Gonzalo Navarro, and Pablo Pérez-Lantero. Maximum-weight planar boxes in $O(n^2)$ time (and better). *Inf. Process. Lett.*, 114(8):437–445, 2014. `doi:10.1016/j.ipl.2014.03.007`.
6. Michael A. Bender and Martin Farach-Colton. The LCA problem revisited. In Gaston H. Gonnet, Daniel Panario, and Alfredo Viola, editors, *Proc. 4th Latin American Symposium on Theoretical Informatics (LATIN)*, volume 1776, pages 88–94, 2000. `doi:10.1007/10719839_9`.
7. Jean-Daniel Boissonnat, Micha Sharir, Boaz Tagansky, and Mariette Yvinec. Voronoi diagrams in higher dimensions under certain polyhedral distance functions. *Discret. Comput. Geom.*, 19(4):485–519, 1998. `doi:10.1007/PL00009366`.

8 Karl Bringmann. An improved algorithm for Klee's measure problem on fat boxes. *Comput. Geom.*, 45(5-6):225–233, 2012. Preliminary version in SoCG'10. doi:10.1016/j.comgeo.2011.12.001.

9 Timothy M. Chan. Geometric applications of a randomized optimization technique. *Discret. Comput. Geom.*, 22(4):547–567, 1999. doi:10.1007/PL00009478.

10 Timothy M. Chan. A (slightly) faster algorithm for Klee's measure problem. *Comput. Geom.*, 43(3):243–250, 2010. Preliminary version in SoCG'08. doi:10.1016/j.comgeo.2009.01.007.

11 Timothy M. Chan. Klee's measure problem made easy. In *Proc. 54th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 410–419, 2013. doi:10.1109/FOCS.2013.51.

12 Timothy M. Chan. (Near-)linear-time randomized algorithms for row minima in Monge partial matrices and related problems. In *Proc. 32nd ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1465–1482, 2021. doi:10.1137/1.9781611976465.88.

13 Timothy M. Chan, John Hershberger, and Simon Pratt. Two approaches to building time-windowed geometric data structures. *Algorithmica*, 81(9):3519–3533, 2019. doi:10.1007/s00453-019-00588-3.

14 Timothy M. Chan and Patrick Lee. On constant factors in comparison-based geometric algorithms and data structures. *Discret. Comput. Geom.*, 53(3):489–513, 2015. doi:10.1007/s00454-015-9677-y.

15 Bernard Chazelle, Robert L. (Scot) Drysdale III, and D. T. Lee. Computing the largest empty rectangle. *SIAM J. Comput.*, 15(1):300–315, 1986. doi:10.1137/0215022.

16 Karen L. Daniels, Victor J. Milenkovic, and Dan Roth. Finding the largest area axis-parallel rectangle in a polygon. *Comput. Geom.*, 7:125–148, 1997. doi:10.1016/0925-7721(95)00041-0.

17 Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Mark H. Overmars. *Computational Geometry: Algorithms and Applications*. Springer, 3rd edition, 2008. URL: https://www.worldcat.org/oclc/227584184.

18 Adrian Dumitrescu and Minghui Jiang. On the largest empty axis-parallel box amidst n points. *Algorithmica*, 66(2):225–248, 2013. doi:10.1007/s00453-012-9635-5.

19 Adrian Dumitrescu and Minghui Jiang. Perfect vector sets, properly overlapping partitions, and largest empty box. *CoRR*, abs/1608.06874, 2016. arXiv:1608.06874.

20 Adrian Dumitrescu and Minghui Jiang. On the number of maximum empty boxes amidst n points. *Discret. Comput. Geom.*, 59(3):742–756, 2018. Preliminary version in SoCG'16. doi:10.1007/s00454-017-9871-1.

21 Adrian Dumitrescu, Joseph S. B. Mitchell, and Micha Sharir. Binary space partitions for axis-parallel segments, rectangles, and hyperrectangles. *Discret. Comput. Geom.*, 31(2):207–227, 2004. doi:10.1007/s00454-003-0729-3.

22 Harold N. Gabow, Jon Louis Bentley, and Robert Endre Tarjan. Scaling and related techniques for geometry problems. In *Proc. 16th ACM Symposium on Theory of Computing (STOC)*, pages 135–143, 1984. doi:10.1145/800057.808675.

23 Panos Giannopoulos, Christian Knauer, Magnus Wahlström, and Daniel Werner. Hardness of discrepancy computation and epsilon-net verification in high dimension. *J. Complex.*, 28(2):162–176, 2012. doi:10.1016/j.jco.2011.09.001.

24 Maria M. Klawe and Daniel J. Kleitman. An almost linear time algorithm for generalized matrix searching. *SIAM J. Discret. Math.*, 3(1):81–97, 1990. doi:10.1137/0403009.

25 Lawrence L. Larmore. An optimal algorithm with unknown time complexity for convex matrix searching. *Inf. Process. Lett.*, 36(3):147–151, 1990. doi:10.1016/0020-0190(90)90084-B.

26 Amnon Naamad, D. T. Lee, and Wen-Lian Hsu. On the maximum empty rectangle problem. *Discret. Appl. Math.*, 8(3):267–277, 1984. doi:10.1016/0166-218X(84)90124-0.

27 Mark H. Overmars and Chee-Keng Yap. New upper bounds in Klee's measure problem. *SIAM J. Comput.*, 20(6):1034–1045, 1991. doi:10.1137/0220065.

28 Seth Pettie and Vijaya Ramachandran. An optimal minimum spanning tree algorithm. *J. ACM*, 49(1):16–34, 2002. doi:10.1145/505241.505243.

29 Micha Sharir and Pankaj K. Agarwal. *Davenport-Schinzel Sequences and Their Geometric Applications*. Cambridge University Press, 1995.

30 Mario Ullrich and Jan Vybíral. An upper bound on the minimal dispersion. *J. Complex.*, 45:120–126, 2018. doi:10.1016/j.jco.2017.11.003.

31 Jean Vuillemin. A unifying look at data structures. *Commun. ACM*, 23(4):229–239, 1980. doi:10.1145/358841.358852.

More Dynamic Data Structures for Geometric Set Cover with Sublinear Update Time

Timothy M. Chan
Department of Computer Science, University of Illinois at Urbana-Champaign, IL, USA

Qizheng He
Department of Computer Science, University of Illinois at Urbana-Champaign, IL, USA

Abstract

We study geometric set cover problems in dynamic settings, allowing insertions and deletions of points and objects. We present the first dynamic data structure that can maintain an $O(1)$-approximation in sublinear update time for set cover for axis-aligned squares in 2D. More precisely, we obtain randomized update time $O(n^{2/3+\delta})$ for an arbitrarily small constant $\delta > 0$. Previously, a dynamic geometric set cover data structure with sublinear update time was known only for unit squares by Agarwal, Chang, Suri, Xiao, and Xue [SoCG 2020]. If only an approximate size of the solution is needed, then we can also obtain sublinear amortized update time for disks in 2D and halfspaces in 3D. As a byproduct, our techniques for dynamic set cover also yield an optimal randomized $O(n \log n)$-time algorithm for static set cover for 2D disks and 3D halfspaces, improving our earlier $O(n \log n (\log \log n)^{O(1)})$ result [SoCG 2020].

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Geometric set cover, approximation algorithms, dynamic data structures, sublinear algorithms, random sampling

Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.25

Related Version *Full Version*: `https://arxiv.org/abs/2103.07857`

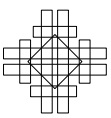
Funding *Timothy M. Chan*: Supported in part by NSF Grant CCF-1814026.

1 Introduction

Approximation algorithms for NP-hard problems and dynamic data structures are two of the major themes studied by the algorithms community. Recently, problems at the intersection of these two threads have gained much attention, and researchers in computational geometry have also started to systematically explore such problems. For example, at SoCG last year, two papers appeared, one on dynamic geometric set cover by Agarwal et al. [2], and another on dynamic geometric independent set by Henzinger, Neumann, and Wiese [19]. In this paper, we continue the study by Agarwal et al. [2] and investigate dynamic data structures for approximating the minimum set cover in natural geometric instances.

Static geometric set cover. In the static (unweighted) *geometric set cover* problem, we are given a set X of $O(n)$ points and a set S of $O(n)$ geometric objects, and we want to find the smallest subset of objects in S that covers all points of X. The problem is fundamental and has many applications. Let OPT denote the value (i.e., cardinality) of the optimal solution. As the problem is NP-hard for many classes of geometric objects, we are interested in efficient approximation algorithms.

Many classes of objects, such as squares and disks in 2D, objects with linear union complexity, and halfspaces in 3D, admit polynomial-time $O(1)$-approximation algorithms (i.e., computing a solution of size $O(1) \cdot \mathrm{OPT}$), by using ε-nets and LP rounding or the



multiplicative weight update (MWU) method [6, 14, 15]. Some classes of objects, such as disks in 2D and halfspaces in 3D, even have PTASs [26] or quasi-PTASs [25], though with large polynomial running time.

Agarwal and Pan [4] worked towards finding *faster* approximation algorithms that run in near linear time. They gave randomized $O(n \log^4 n)$-time algorithms (based on MWU) for computing $O(1)$-approximations, for example, for disks in 2D and halfspaces in 3D. At last year's SoCG [11], we described further improvements to the running time for 2D disks and 3D halfspaces, including a deterministic $O(n \log^3 n \log\log n)$-time algorithm and a randomized $O(n \log n (\log\log n)^{O(1)})$-time algorithm.

Dynamic geometric set cover. It is natural to consider the *dynamic* setting of the geometric set cover problem. Here, we want to support insertions and deletions of points in X *as well as* insertions and deletions of objects in S, while maintaining an approximate solution. Note that the solution may have linear size in the worst case. In the simplest version of the problem, we may just want to output the value of the solution. More strongly, we may want some representation of the solution itself, so that afterwards, the objects in the solution can be reported in constant time per element when needed.

Agarwal et al. [2] gave a number of results on dynamic geometric set cover. They showed that for intervals in 1D, a $(1+\varepsilon)$-approximation can be maintained in $O((1/\varepsilon)n^{\delta})$ time per insertions and deletions of points and intervals. (Throughout the paper, $\delta > 0$ denotes an arbitrarily small constant.) In 2D, they had only one main result: a fully dynamic $O(1)$-approximation algorithm for unit axis-aligned squares, with $O(n^{1/2+\delta})$ update time.

New results. We present several new results on dynamic geometric set cover. The first is a fully dynamic, randomized, $O(1)$-approximation algorithm for the more general case of arbitrary axis-aligned squares in 2D, with $O(n^{2/3+\delta})$ amortized update time. Though our time bound is a little worse than Agarwal et al.'s for the unit square case, the arbitrary square case is more challenging. (The unit square case reduces to case of dominance ranges, i.e., quadrants, via a standard grid approach; since the union of such ranges forms a "staircase" sequence of vertices, the problem is in some sense "1.5-dimensional". In contrast, the arbitrary square problem requires truly "2-dimensional" ideas.)

We then consider the case of halfspaces in 3D. This case is fundamental, as set cover for 2D disks reduces to set cover for 3D halfspaces by the standard lifting transformation [17]. Also, by duality, hitting set for 3D halfspaces is equivalent to set cover for 3D halfspaces, and hitting set for 2D disks reduces to set cover for 3D halfspaces as well.

For 3D halfspaces, we obtain a fully dynamic, randomized algorithm with $O(n^{12/13+\delta}) \leq O(n^{0.924})$ amortized update time. Our result here is slightly weaker: it only finds the value of an $O(1)$-approximate solution (which could be good enough in some applications). If a solution itself is required, we can still get sublinear update time as long as OPT is sublinear (below $n^{1-\delta}$). This assumption seems reasonable, since sublinear reporting time is not possible otherwise. (However, we currently do not know how to obtain a stronger time bound of the form $O(n^{\alpha} + \mathrm{OPT})$ with $\alpha < 1$ to report a solution for 3D halfspaces.)

Remarks. Our results are randomized in the Monte Carlo sense: the computed solution may not always be correct, but it is correct with high probability (w.h.p.), i.e., probability at least $1 - 1/n^c$ for an arbitrarily large constant c. The error probability bounds hold even when the user has knowledge of the random choices made by the algorithms. (We do not assume an "oblivious adversary"; in fact, the algorithms make a new set of random choices each time it computes a solution, and the data structures themselves are not randomized.)

The update times are better than stated in many cases, notably, when OPT is small or when OPT is large. More precise OPT-sensitive bounds are given in lemmas and theorems throughout the paper. (The $O(n^{2/3+\delta})$ and $O(n^{12/13+\delta})$ bounds are obtained by "balancing".)

We have assumed that we are required to compute a solution after every update. In some (but not all) cases, the update cost is smaller than the cost of computing a solution (this may be useful if we are executing a batch of updates).

Techniques. Our algorithms are obtained by handling two cases differently: when OPT is small and when OPT is large. Intuitively, the small OPT case is easier since we are generating fewer objects, but the large OPT case also seems potentially easier since we can tolerate a larger additive error when targeting an $O(1)$-factor approximation – so, we are in a "win-win" situation. For our algorithms for 3D halfspaces, we even find it necessary to handle an intermediate case when OPT is medium (aiming for sublinear time for sublinear OPT).

Our algorithms for the small OPT case are based on the previous static MWU algorithms [6, 4, 11]. The adaptation of these static algorithms is not straightforward, and requires using various known techniques in new ways ($(\leq k)$-levels in arrangements, for our 2D square algorithm in Section 3.1, and "augmented" partition trees, for our 3D halfspace algorithm in Section 4.1). The medium case for 3D halfspaces (in Section 4.2) is technically even more challenging (where we use "shallow" partition trees and other ideas).

Our algorithm for squares in the large OPT case (in Section 3.2) is different (not based on MWU), and interestingly uses quadtrees in a non-obvious way. For 3D halfspaces, our algorithm in the large OPT case (in Section 4.3) can compute only the value of an approximate solution, and is based on random sampling. However, the obvious way to use a random sample (just solving the problem on a random subset of points and objects) does not work. We use sampling in a nontrivial way, combining geometric cuttings with planar graph separators.

In some of our algorithms, notably the small OPT algorithms for squares and 3D halfspaces (in Sections 3.1 and 4.1) and the large OPT algorithm for 3D halfspaces (in Section 4.3), the data structure part is "minimal": we just assume that points and objects are stored separately in standard range searching data structures. We describe sublinear-time algorithms to compute a solution from scratch, using range searching as oracles. Dynamization becomes trivial, since range searching data structures typically are already known to support insertions and deletions. (It also potentially enables other operations like merging sets, or solving the set cover problem for range-restricted subsets of points and subsets of objects.)

The topic of *sublinear-time algorithms* has received considerable attention in the algorithms community, due to applications to big data (where we want to solve problems without examining the entire input). A similar model of sublinear-time algorithms where the input is augmented with range searching data structures was proposed by Czumaj et al. [16], who presented results on approximating the weight of the Euclidean minimum spanning tree in any constant dimension under this model.

Application to static geometric set cover. Although we did not intend to revisit the static problem, our techniques can lead a randomized $O(1)$-approximation algorithm for set cover for 3D halfspaces running in $O(n \log n)$ time, which completely eliminates the extra $\log \log n$ factors in our previous result from SoCG'20 [11] and is optimal (in comparison-based models)! This bonus result is interesting in its own right, but due to lack of space, we defer the description to the full paper.

2 Review of an MWU algorithm

We begin by briefly reviewing a known static approximation algorithm for geometric set cover, based on the *multiplicative weight updates (MWU)* method. Some of our dynamic algorithms will be built upon this algorithm.

Specifically, we consider the following randomized algorithm from our previous SoCG'20 paper [11], which is a variant of a standard algorithm by Brönnimann and Goodrich [6] or Clarkson [14] (see also Agarwal and Pan [4]). Below, c_0 is a sufficiently large constant. The *depth* of a point p in a set S of objects is the number of objects of S containing p. A subset of objects $T \subseteq S$ is called an *ε-net* of S if all points with depth $\geq \varepsilon|S|$ in S are covered by T. The size $|\hat{S}|$ of a multiset $\hat{S}$ refers to the sum of the multiplicities of its elements $\sum_{i \in S} m_i$.

Algorithm 1 MWU for set cover.

1: Guess a value $t \in [\mathrm{OPT}, 2\,\mathrm{OPT}]$.
2: Define a multiset $\hat{S}$ where each object i in S initially has multiplicity $m_i = 1$.
3: **loop** ▷ call this the start of a new *round*
4: Fix $\rho := \frac{c_0 t \log n}{|\hat{S}|}$ and take a random sample R of $\hat{S}$ with sampling probability ρ.
5: **while** there exists a point $p \in X$ with depth in R at most $\frac{c_0}{2} \log n$ **do**
6: **for** each object i containing p **do** ▷ call lines 6–8 a *multiplicity-doubling step*
7: Double its multiplicity m_i, i.e., insert m_i new copies of object i into $\hat{S}$.
8: For each copy, independently decide to insert it into R with probability ρ.
9: **if** the number of multiplicity-doubling steps in this round exceeds t **then**
10: Go to line 3 and start a new round.
11: Terminate and return a $\frac{1}{8t}$-net of R.

In the standard version of MWU, the "lightness" condition in line 5 was whether the depth of p in $\hat{S}$ is at most $\frac{|\hat{S}|}{2t}$. The main difference in the above randomized version is that lightness is tested with respect to the sample R, which is computationally easier to work with – R has size $O(t \log n)$ with high probability (w.h.p.). Justification of this randomized variant follows from a Chernoff bound, and was shown in [11, Section 4.2].

It is known that the algorithm terminates in $O(\log \frac{n}{t})$ rounds and $O(t \log \frac{n}{t})$ multiplicity-doubling steps [6, 4, 11]. Furthermore, $|\hat{S}|$ increases by at most a factor of 2 in each round; in particular, $|\hat{S}|$ is bounded by $n^{O(1)}$ at the end of $O(\log \frac{n}{t})$ rounds.

In the standard version of MWU, line 11 returns a $\Theta(\frac{1}{t})$-net of the multiset $\hat{S}$, but a $\Theta(\frac{1}{t})$-net of R works just as well: in the end, the depth in R of all points of X is at least $\frac{c_0}{2} \log n > \frac{|R|}{8t}$ w.h.p., and so X will be covered by the net. For objects that are axis-aligned squares in 2D, disks in 2D, or halfspaces in 3D, ε-nets of size $O(\frac{1}{\varepsilon})$ exist, and thus the above algorithm yields a set cover of size $O(t)$, i.e., a constant-factor approximation. By known algorithms (e.g., [12]), the net for R in line 11 can be constructed in $\tilde{O}(|R|) = \tilde{O}(t)$ time.

Several modifications have been explored in previous work. For example, in the first algorithm of Agarwal and Pan [4], each round examines all points of X in a fixed order and test for lightness of the points one by one (based on the observation that a point found to have large depth will still have large depth by the end of the round). In our previous paper [11], we have also added a step at the beginning of each round, where the multiplicities are rescaled and rounded, so as to keep $|\hat{S}|$ bounded by $O(n)$. These modifications led to a number of different static implementations running in $\tilde{O}(n)$ time.

3 Axis-aligned squares

Our first result for dynamic set cover is for axis-aligned squares. Previously, a sublinear time algorithm for dynamic set cover was known only for unit squares. Our method will be divided into two cases: when OPT is small and when OPT is large. Let t be a guess on OPT. We will run our algorithm for each possible $t = 2^i$ in parallel. (When the guess is wrong, our algorithm will be able to tell whether t is approximately smaller or larger than OPT w.h.p.) The update time will increase only by a factor of $O(\log n)$.

3.1 Algorithm for small OPT

Our algorithm for the small OPT case will be based on the randomized MWU algorithm described in Section 2. The key is to realize that this algorithm can actually be implemented to run in *sublinear* time, assuming that the points and the objects have been preprocessed in standard range searching data structures. Since these structures are dynamizable, we can just re-run the MWU algorithm from scratch after every update.

3.1.1 Data structures

Our data structures are simple. We store the point set X in the standard 2D *range tree* [17]. For each square s with center (x, y) and side length $2z$, map s to a point $s^{\uparrow} = (x - z, x + z, y - z, y + z)$ in 4D. We also store the lifted point set $S^{\uparrow} = \{s^{\uparrow} : s \in S\}$ in a 4D range tree. Range trees support insertions and deletions in X and S in polylogarithmic time.

3.1.2 Computing a solution

We now show how to compute an $O(1)$-approximate solution in sublinear time when OPT is small by running Algorithm 1 using the above data structures. At first glance, linear time seems unavoidable: (i) the obvious way to find low-depth points in line 5 is to scan through all points of X in each round (as was done in previous algorithms [4, 11]), and (ii) explicitly maintaining the multiplicities of all points would also require linear time.

To overcome these obstacles, we observe that (i) we can use data structures to find the next low-depth point p without testing each point one by one (recall that there are only $\tilde{O}(t)$ multiplicity-doubling steps), and (ii) multiplicities do not need to be maintained explicitly, so long as in line 8 we can generate a multiplicity-weighted random sample among the objects containing a given point p efficiently (recall that the sample R has only $\tilde{O}(t)$ size). The subproblem in (ii) is a *weighted range sampling* problem.

Finding a low-depth point. Let $b := \frac{c_0}{2} \log n$. Each time we want to find a low-depth point in line 5, we compute (from scratch) $\mathcal{L}_{\le b}(R)$, the $(\le b)$-level of R, i.e., the collection of all cells in the arrangement of the squares of depth at most b. It is known [27] that $\mathcal{L}_{\le b}(R)$ has $O(|R|b)$ cells and can be constructed in $\tilde{O}(|R|b)$ time, which is $\tilde{O}(t)$ (since $|R| = \tilde{O}(t)$ and $b = \tilde{O}(1)$). To find a point p of X that has depth in R at most b, we simply examine each cell of $\mathcal{L}_{\le b}(R)$, and perform an orthogonal range query to test if the cell contains a point of X. All this takes $\tilde{O}(t)$ time.

As there are $\tilde{O}(t)$ multiplicity-doubling steps, the total cost is $\tilde{O}(t^2)$.

Weighted range sampling. For each square s with center (x, y) and side length $2z$, define its *dual* point s^* to be (x, y, z) in 3D. For each point $p = (p_x, p_y)$, define its *dual* region p^* to be $\{(x, y, z) : z \ge \max\{|x - p_x|, |y - p_y|\}\}$ in 3D. Then a point p is in the square s iff the point s^* is in the region p^*.

Let Q be the set of all points p for which we have performed multiplicity-doubling steps thus far. Note that $|Q| = \tilde{O}(t)$. Let $b' = c'_0 \log n$ for a sufficiently large constant c'_0. Each time we perform a multiplicity-doubling step, we compute (from scratch) $\mathcal{L}_{\leq b'}(Q^*)$, the $(\leq b')$-level of the dual regions $Q^* = \{p^* : p \in Q\}$. This structure corresponds to planar order-$(\leq b')$ L_∞ Voronoi diagrams. By known results [27, 7, 21], $\mathcal{L}_{\leq b'}(Q^*)$ has $O(|Q|(b')^2)$ cells and can be constructed in $\tilde{O}(|Q|(b')^2)$ time, which is $\tilde{O}(t)$ (since $|Q| = \tilde{O}(t)$ and $b' = \tilde{O}(1)$). The multiplicity of a square $s \in S$ is equal to $2^{\text{depth of } s^* \text{ in } Q^*}$ (since each p^* in Q^* containing s^* doubles the multiplicity of s). In particular, since multiplicities are bounded by $n^{O(1)}$, the depth (i.e., level) of s^* must be logarithmically bounded. So, each s^* is covered by $\mathcal{L}_{\leq b'}(Q^*)$, and the multiplicity of s is determined by which cell of $\mathcal{L}_{\leq b'}(Q^*)$ the point s^* is in.

To generate a multiplicity-weighted sample of the squares containing p for line 8, after p has been inserted to Q, we examine all cells of $\mathcal{L}_{\leq b'}(Q^*)$ contained in p^*. For each such cell γ, we identify the squares $s \in S$ for which $s^* \in \gamma$; this reduces to an orthogonal range query in $S^\uparrow$, and the answer can be expressed as a disjoint union of $\tilde{O}(1)$ canonical subsets. Knowing the sizes and multiplicities of these canonical subsets for all such $\tilde{O}(t)$ cells, we can then generate the weighted sample in time $\tilde{O}(t)$ plus the size of the sample.

Hence, the total cost of all $\tilde{O}(t)$ multiplicity-doubling steps is $\tilde{O}(t^2)$.

In addition, we need to generate a new weighted sample R (with a new sampling probability ρ) in line 4 at the beginning of each round; this can be done similar to above, in $\tilde{O}(t)$ time plus the size of the sample ($\tilde{O}(t)$), for each of the $O(\log n)$ rounds. As mentioned, the final net computation in line 11 takes $\tilde{O}(t)$ time. We have thus obtained:

▶ **Lemma 1.** *There exists a data structure for the dynamic set cover problem for $O(n)$ axis-aligned squares and $O(n)$ points in 2D that supports insertions and deletions in $\tilde{O}(1)$ time and can find an $O(1)$-approximate solution w.h.p. to the set cover problem in $\tilde{O}(\mathrm{OPT}^2)$ time.*

3.2 Algorithm for large OPT

To complement our solution for the small OPT case, we now show that the problem also gets easier when OPT is large, mainly because we can afford a large additive error. We describe a different, self-contained algorithm for this case (not based on modifying MWU), interestingly by using quadtrees in a novel way.

To allow for both multiplicative and additive error, we use the term (α, β)-*approximation* to refer to a solution with cost at most $\alpha\,\mathrm{OPT} + \beta$.

For simplicity, we assume that all coordinates are integers bounded by $U = \mathrm{poly}(n)$. At the end, we will comment on how to remove this assumption.

In the standard *quadtree*, we start with a bounding square cell and recursively divide a square cell into four square subcells. We define the size of a cell Γ to be the number of vertices in Γ among the squares of S, plus the number of points of X in Γ. We stop subdividing when a leaf cell has size at most b, where b is a parameter to be set later. This yields a subdivision into $O(\frac{n}{b})$ cells per level, and $O(\frac{n}{b}\log U)$ cells in total. The quadtree decomposition can be easily made dynamic under insertions and deletions of points and squares.

For each leaf cell Γ, a square in S intersecting Γ is called *short* if at least one of its vertices is in the cell, and *long* otherwise, as shown in Fig. 1. Note that a long square can have at most one side crossing the cell, because the quadtree cell Γ is also a square. The union of the long squares within the cell is defined by at most 4 long squares – call these the *maximal* long squares. (If there is a square containing Γ, we can designate one such square as the maximal long square.) For each leaf cell Γ, it suffices to approximate the optimal set cover for the input points in $X \cap \Gamma$ using only the short squares plus the at most 4 maximal long

squares in Γ. By charging each square in the optimal solution to the cells containing its 4 vertices, we see that the sum of the sizes of the optimal covers in the leaf cells is at most $4\,\mathrm{OPT} + O(\frac{n}{b}\log U)$, which is indeed an $O(1)$-approximation if we choose $b \geq \frac{n \log n}{\mathrm{OPT}}$.

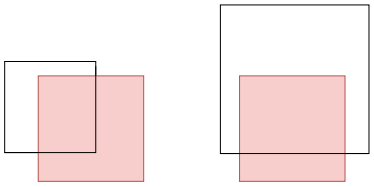


Figure 1 Short square (left) and long square (right). The quadtree cell is shaded.

Note that the complement of the union of the at most 4 maximal long squares in a cell Γ is a rectangle r_Γ. We will store the short squares in the cell Γ in a data structure $\mathcal{S}_\Gamma$ to answer the following type of query:

> Given any query rectangle r, compute an $O(1)$-approximation to the optimal set cover for the points in $X \cap r$ using only the short squares.

Assuming the availability of such data structures $\mathcal{S}_\Gamma$, we can solve the dynamic set cover problem as follows:

- An insertion/deletion of a square s in S requires updating 4 of these data structures $\mathcal{S}_\Gamma$ for the leaf cells Γ containing the 4 vertices of s, and also updating the maximal long squares for all leaf cells in $\tilde{O}(\frac{n}{b})$ time.
- An insertion/deletion of a point p in X requires updating one data structure $\mathcal{S}_\Gamma$ for the leaf cell Γ containing p.
- Whenever we want to compute a set cover solution, we examine all $\tilde{O}(\frac{n}{b})$ cells Γ and query the data structure $\mathcal{S}_\Gamma$ for r_Γ, and return the union of the answers.

First implementation of $\mathcal{S}_\Gamma$. A simple way to implement the data structure $\mathcal{S}_\Gamma$ is as follows: in an update, we just recompute an approximate solution from scratch for every possible query rectangle in Γ. Since there are only $O(b^4)$ combinatorially different query rectangles, and static approximate set cover on $O(b)$ squares and points takes $\tilde{O}(b)$ time [4, 11], the update time is $\tilde{O}(b^5)$, and the query time is trivially $O(1)$.

As a result, the cost of insertion/deletion in the overall method is $\tilde{O}(b^5 + \frac{n}{b})$.

Improved implementation for $\mathcal{S}_\Gamma$. We further improve the update time for the data structure $\mathcal{S}_\Gamma$. Instead of recomputing solutions for all $O(b^4)$ rectangles, the idea is to recompute solutions for a smaller number of "canonical rectangles". More precisely, by using a 2D range tree [3, 17] for the $O(b)$ points in $X \cap \Gamma$, with branching factor a in each dimension, we can form a set of canonical rectangles with total size $O(a^{O(1)} b(\log_a b)^2)$, such that every query rectangle can be decomposed into $O((\log_a b)^2)$ canonical rectangles, ignoring portions that are empty of points. We set $a := b^\delta$ for an arbitrarily small constant $\delta > 0$.

For each canonical rectangle r with size b_i, there are at most $O(b_i)$ maximal long squares with respect to r: the union of the long squares that cut across r horizontally have at most two edges between any two consecutive points/vertices; and a similar statement holds for the long squares that cut across r vertically. These maximal long squares can be found in $\tilde{O}(b_i)$ time by standard orthogonal range searching. We can thus approximate the optimal set cover for the points in the canonical rectangle r, using the $O(b_i)$ short squares and maximal long squares with respect to r, in $\tilde{O}(b_i)$ time by known static set cover algorithms [4, 11]. The total time over all canonical rectangles is $\tilde{O}(a^{O(1)} b(\log_a b)^2) = \tilde{O}(b^{1+O(\delta)})$.

Given a query rectangle, we can decompose it into $O((\log_a b)^2) = O(1)$ canonical rectangles and return the union of the optimal solutions in the canonical rectangles, which is an $O(1)$-approximation (more precisely, an $O(\delta^{-2})$-approximation).

As a result, the cost of insertion/deletion in the overall method is $\tilde{O}(b^{1+O(\delta)} + \frac{n}{b})$.

Removing the dependency on U. When U may be large, we can reduce the tree depth from $O(\log U)$ to $O(\log n)$ by replacing the quadtree with the *BBD tree* of Arya et al. [5]. Each cell in the BBD tree is the set difference of two quadtree squares, one contained in the other. The size of each child cell is at most a fraction of the size of the parent cell. As before, we stop subdividing when a leaf cell has size at most b. Since any such leaf cell has size $\Theta(b)$ now, the number of leaf cells is $O(\frac{n}{b})$. The BBD tree can be maintained dynamically in polylogarithmic time (for example, by periodically rebuilding when subtrees become unbalanced). Since a leaf cell Γ is the difference of two quadtree squares, it is not difficult to see that the number of maximal long squares in Γ remains $O(1)$. So, our previous analysis remains valid.

▶ **Lemma 2.** *Given a parameter b, there exists a data structure for the dynamic set cover problem for $O(n)$ axis-aligned squares and $O(n)$ points in 2D that maintains an $(O(1), O(\frac{n}{b}))$-approximate solution with $\tilde{O}(b^{1+O(\delta)} + \frac{n}{b})$ insertion and deletion time.*

Combining the algorithms. When $\mathrm{OPT} \le n^{1/3}$, we use the algorithm for small OPT; the running time is $\tilde{O}(\mathrm{OPT}^2) \le \tilde{O}(n^{2/3})$. When $\mathrm{OPT} > n^{1/3}$, we use the algorithm for large OPT with $b = n^{2/3}$, so that an $(O(1), O(\frac{n}{b}))$-approximation is indeed an $O(1)$-approximation; the running time is $\tilde{O}(b^{1+O(\delta)} + \frac{n}{b}) = O(n^{2/3+O(\delta)})$.

▶ **Theorem 3.** *There exists a data structure for the dynamic set cover problem for $O(n)$ axis-aligned squares and $O(n)$ points in 2D that maintains an $O(1)$-approximate solution w.h.p. with $O(n^{2/3+\delta})$ insertion and deletion time for any constant $\delta > 0$.*

The case of fat rectangles can be reduced to squares, since such rectangles can be replaced by $O(1)$ squares (increasing the approximation factor by only $O(1)$). Our approach can be modified to work more generally for homothets of a fixed fat convex polygon with a constant number of vertices.

4 Halfspaces in 3D

In this section, we study dynamic geometric set cover for the more challenging case of 3D halfspaces. Using the standard lifting transformation [17], we can transform 2D disks to 3D upper halfspaces. For simplicity, we assume that all halfspaces are upper halfspaces; we discuss how to modify our algorithms when there are both upper and lower halfspaces in the full paper. Our method will be divided into three cases: small, medium, and large OPT.

4.1 Algorithm for small OPT

Similar to the small OPT algorithm for axis-aligned squares in Section 3.1, we describe a small OPT algorithm for halfspaces based on the randomized MWU algorithm in Section 2. Although our earlier approach using levels in arrangements could be generalized, we describe a better approach based on augmenting partition trees with counters.

4.1.1 Data structures

We store the 3D point set X in Matoušek's *partition tree* [22]: The tree has height $O(\log n)$ and degree r for a sufficiently large constant r. Each node v stores a simplicial cell Γ_v and a "canonical subset" $X_v \subset \Gamma_v$, where $X_v = X$ at the root v, and Γ_v is contained in $\Gamma_{\text{parent}(v)}$, X_v is the disjoint union of $X_{v'}$ over all children v' or v, and X_v has constant size at each leaf v. Furthermore, any halfspace crosses $O(n^{2/3+\delta})$ cells of the tree for any arbitrarily small constant $\delta > 0$ (depending on r). Here a halfspace h *crosses* a cell Γ iff the boundary of h intersects Γ.

For each upper halfspace h, let h^* denote its dual point; for each point p, let p^* denote its dual upper halfspace. (Duality [17] is defined so that p is in h iff h^* is in p^*.)

We also store the 3D dual point set $S^* = \{h^* : h \in S\}$ in Matoušek's partition tree. Each node v stores a cell Γ_v and a canonical subset $S^*_v \subset \Gamma_v$ like above.

Matoušek's partition trees can be built in $\tilde{O}(n)$ time and support insertions and deletions in X and S in polylogarithmic time. (In the static case, there are slightly improved partition trees reducing the n^δ factor in the crossing number bound [22, 24, 10], but these will not be important to us.)

4.1.2 Computing a solution

We now show how to compute an $O(1)$-approximate solution in sublinear time when OPT is small by running Algorithm 1 using the above data structures. As in Section 3.1, the main subproblems are (i) finding a low-depth point with respect to R, and (ii) weighted range sampling, where the weights are the multiplicities (which are not explicitly stored).

Finding a low-depth point. We maintain two values at each node v of the partition tree of X:

- c_v is the number of halfspaces of R containing Γ_v but not containing $\Gamma_{\text{parent}(v)}$.
- d_v is the minimum depth among all points in X_v with respect to the halfspaces of R crossing Γ_v.

The overall minimum depth with respect to R is given by the value d_v at the root v. Whenever we insert a halfspace h to R, for each of the $O(n^{2/3+\delta})$ cells Γ_v crossed by h, we update the counters $c_{v'}$ for the children v' of the nodes v; we also update the value d_v bottom-up according to the formula $d_v = \min_{\text{child } v' \text{ of } v}(d_{v'} + c_{v'})$. Thus, all values can be maintained in $O(n^{2/3+\delta})$ time per insertion to R.

As there are $\tilde{O}(t)$ insertions to R, the total cost is $\tilde{O}(tn^{2/3+\delta})$. This cost covers the resetting of counters at every round.

(We remark that the idea of augmenting nodes of partition trees with counters appeared before in at least one prior work on dynamic geometric data structures [8, Theorem 4.1].)

Weighted range sampling. Let Q be the set of all points p for which we have performed multiplicity-doubling steps thus far. Note that $|Q| = \tilde{O}(t)$. The multiplicity of a halfspace $h \in S$ is $2^{\text{depth of } h^* \text{ in } Q^*}$. To implicitly represent the multiplicities and their sum, we maintain two values at each node v of the partition tree for S^*:

- c_v is the number of dual halfspaces of Q^* containing Γ_v but not containing $\Gamma_{\text{parent}(v)}$.
- m_v is the sum of $2^{\text{depth of } h^* \text{ among the halfspaces of } Q^* \text{ crossing } \Gamma_v}$ over all $h^* \in S^*_v$.

Whenever we insert a point p to Q, for each of the $O(n^{2/3+\delta})$ cells Γ_v crossed by the dual halfspace p^*, we update the counters $c_{v'}$ for the children v' of the nodes v; we also update the value m_v bottom-up according to the formula $m_v = \sum_{\text{child } v' \text{ of } v} 2^{c_{v'}} m_{v'}$. Thus, all values can be maintained in $O(n^{2/3+\delta})$ time per insertion to Q.

To generate a weighted sample of the halfspaces of S containing p for line 8, we find all $O(n^{2/3+\delta})$ cells Γ_v crossed by the dual halfspace p^*, and consider the canonical subsets $S^*_{v'}$ for the children v' of v with $\Gamma_{v'}$ contained in p^*. We can then sample from these canonical subsets, weighted by $m_{v'}2^{\sum_u c_u}$, where the sum is over all ancestors u of v'. All this takes time $O(n^{2/3+\delta})$ plus the size of the sample.

Hence, the total cost of all $\tilde{O}(t)$ multiplicity-doubling steps is $\tilde{O}(tn^{2/3+\delta})$.

▶ **Lemma 4.** *There exists a data structure for the dynamic set cover problem for $O(n)$ upper halfspaces and $O(n)$ points in 3D that supports insertions and deletions in $\tilde{O}(1)$ time and can find an $O(1)$-approximate solution w.h.p. in $\tilde{O}(\mathrm{OPT} \cdot n^{2/3+\delta})$ time.*

4.2 Algorithm for medium OPT

The preceding algorithm works well only when OPT is smaller than about $n^{1/3}$. We show that a more involved algorithm, also based on MWU, can achieve sublinear time even when OPT approaches $n^{1-\delta}$. The basic approach is to use *shallow* versions of the partition trees [23]. Several technical new ideas are required, but due to space limitation, we defer the details to the full paper and just state the result:

▶ **Lemma 5.** *There exists a data structure for the dynamic set cover problem for $O(n)$ upper halfspaces and $O(n)$ points in 3D that maintains an $O(1)$-approximate solution w.h.p. with $\tilde{O}(\frac{n^{7/8}}{\mathrm{OPT}^{1/8}} + \frac{n}{\sqrt{\mathrm{OPT}}} + \mathrm{OPT}^{1/3}n^{2/3+O(\delta)})$ amortized insertion and deletion time.*

4.3 Algorithm for large OPT

Lastly, we give an algorithm for the large OPT case, which is very different from the algorithms in the previous subsections (and not based on modifying MWU). Here, we can only compute the size of the approximate set cover, not the cover itself. Like before, we will show that the problem gets easier for large OPT, because we can afford a large additive error. The idea is to decompose the problem into subproblems via geometric sampling and planar separators, and then approximate the sum of the subproblems' answers by sampling again.

4.3.1 Data structures

We just store the dual point set S^* in a known 3D halfspace range reporting structure. The data structure by Chan [9] supports queries in $O((\log n + k)\log n)$ time for output size k, and insertions and deletions in S in polylogarithmic amortized time.

We store the xy-projection of the point set X in a known 2D triangle range searching structure [22] that supports queries in $O(\frac{n^{1/2+\delta}}{z^{1/2}} + k)$ time for output size k, and insertions and deletions in X in $\tilde{O}(z)$ time for a given trade-off parameter $z \in [1, n]$.

4.3.2 Approximating the optimal value

Let b and g be parameters to be set later. Take a random sample R of the halfspaces S with size $\frac{n}{b}$. Imagine that R is included in the solution. The remaining uncovered space is the complement of the union of R, which is a 3D convex polyhedron. There are $O(|R|) = O(\frac{n}{b})$ cells in the vertical decomposition $\mathrm{VD}(R)$ of this polyhedron (formed by triangulating each face and drawing a vertical wall at each edge of the triangulation). Each cell is crossed by $O(b \log n)$ halfspaces w.h.p., by well-known geometric sampling analysis [13]. The decomposition $\mathrm{VD}(R)$ can be constructed in $\tilde{O}(\frac{n}{b})$ time.

Our key idea is to use planar graph separators to divide into smaller subproblems. The following is a multi-cluster version of the standard planar separator theorem [20] (sometimes known as "r-divisions" [18]):

▶ **Lemma 6** (Planar Separator Theorem, Multi-Cluster Version). *Given a planar graph $G = (V, E)$ with n vertices, and a parameter g, we can partition V into $\frac{n}{g}$ subsets $V_1, \cdots, V_{n/g}$ of size $O(g)$ each, and an extra "boundary set" B of size $O(\frac{n}{\sqrt{g}})$, such that no two vertices from different subsets V_i and V_j are adjacent. The partition can be constructed in $\tilde{O}(n)$ time.*

(We remark that the general idea of combining cuttings/geometric sampling with planar graph separators appeared in some geometric approximation algorithms before, e.g., [1].)

We apply Lemma 6 to the dual graph of $\mathrm{VD}(R)$ (which has size $O(\frac{n}{b})$), yielding $O((n/b)/g)$ "clusters" of $O(g)$ cells each, and a set B of $O((n/b)/\sqrt{g})$ "boundary cells", in $\tilde{O}(\frac{n}{b})$ time.

Let S_B be the subset of all halfspaces of S that cross boundary cells of B. Note that $|S_B| = O((n/b)/\sqrt{g} \cdot b \log n) = \tilde{O}(\frac{n}{\sqrt{g}})$ w.h.p.

For each cluster γ, let X_γ denote the subset of all points of X whose xy-projections lie in the xy-projection of the cells of γ, and let S_γ denote the subset of all halfspaces of S that cross the cells of γ. Note that $|S_\gamma| = O(g \cdot b \log n) = \tilde{O}(bg)$ w.h.p. Let OPT_γ denote the optimal value for the set cover problem for the halfspaces of S_γ and the points of X_γ not covered by $R \cup S_B$.

▷ Claim 7. $\sum_\gamma \mathrm{OPT}_\gamma$ approximates OPT with additive error $\tilde{O}(\frac{n}{b} + \frac{n}{\sqrt{g}})$ w.h.p.

Proof. A feasible solution can be formed by taking the union of the solutions corresponding to OPT_γ, together with R (to cover points not covered by $\mathrm{VD}(R)$) and S_B (to cover points inside boundary cells). As $|R| = \frac{n}{b}$ and $|S_B| = \tilde{O}(\frac{n}{\sqrt{g}})$ w.h.p., this proves that $\mathrm{OPT} \le \sum_\gamma \mathrm{OPT}_\gamma + \tilde{O}(\frac{n}{b} + \frac{n}{\sqrt{g}})$.

In the other direction, observe that if a halfspace h crosses two different clusters γ_i and γ_j, it must also cross some boundary cell in X by convexity: pick points $p \in h \cap \gamma_i$ and $q \in h \cap \gamma_j$; then the line segment $\overline{pq}$ must hit the wall of some boundary cell. So, after removing $R \cup S_B$ from the global optimal solution, we get disjoint local solutions in the clusters. This proves that $\mathrm{OPT} \ge \sum_\gamma \mathrm{OPT}_\gamma$. ◁

We use the following known fact about approximating a sum via random sampling (which is of course a standard trick):

▶ **Lemma 8.** *Suppose $a_1 + \cdots + a_m = T$ where $a_i \in [0, U]$. Take a random subset R of $r = \frac{(c_0/\varepsilon^2) m U \log n}{T}$ elements from $a_1, \ldots, a_m$, for a sufficiently large constant c_0. Then $\sum_{a_i \in R} a_i \cdot \frac{r}{m}$ is a $(1+\varepsilon)$-approximation to T w.h.p.*

Proof. By rescaling the a_i's and T by a factor U, we may assume that $U = 1$. Define a random variable Y_i, which is a_i with probability $\frac{r}{m}$, and 0 otherwise. Then $E[\sum_i Y_i] = T \cdot \frac{r}{m} = (c_0/\varepsilon^2) \log n$. The result follows from a standard Chernoff bound on the Y_i's. ◀

By applying the above lemma with $m = (n/b)/g$, $T = \Theta(t)$, and $U = \tilde{O}(bg)$ (assuming OPT is finite), we can $O(1)$-approximate OPT by summing OPT_γ over a random sample of $r = \tilde{O}(\frac{mU}{T}) = \tilde{O}(\frac{n}{t})$ clusters γ.

We can generate the set S_B by finding the halfspaces of S that contain the $O((n/b)/\sqrt{g})$ vertices of the cells in B – this corresponds to $O((n/b)/\sqrt{g})$ halfspace range reporting queries for the dual 3D point set S^*, each with output size $\tilde{O}(b)$ w.h.p. and each taking $\tilde{O}(b)$ time [9]. Thus, S_B can be found in $\tilde{O}(\frac{n}{\sqrt{g}})$ time. We compute the union of $R \cup S_B$, which is the complement of an intersection of halfspaces, by the dual of 3D convex hull algorithm [17]. This takes $\tilde{O}(\frac{n}{b} + \frac{n}{\sqrt{g}})$ time.

For each chosen cluster γ, we can generate S_γ similarly by $O(g)$ halfspace range reporting queries for S^*, each with output size $\tilde{O}(b)$ w.h.p. Thus, S_γ can be found in $\tilde{O}(bg)$ time. We can generate X_γ by performing $O(g)$ triangle range reporting queries for the 2D xy-projection of the point set X. Thus, X_γ can be found in $\tilde{O}(g\frac{n^{1/2+\delta}}{z^{1/2}} + |X_\gamma|)$ time. We filter points of X_γ covered by $R \cup S_B$, by performing $|X_\gamma|$ planar point location queries [17] in the xy-projection of the boundary of the union of $R \cup S_B$. This takes $\tilde{O}(|X_\gamma|)$ time. We can then compute an $O(1)$-approximation to OPT_γ by running a known static set cover algorithm [4, 11] in $\tilde{O}(bg + |X_\gamma|)$ time.

The expected sum of X_γ over all chosen clusters γ is $O(n \cdot \frac{r}{m}) = O(rbg)$. The total expected time over r clusters is $\tilde{O}(rbg + rg\frac{n^{1/2+\delta}}{z^{1/2}}) = \tilde{O}(\frac{bgn}{t} + \frac{gn^{3/2+\delta}}{tz^{1/2}})$. The overall expected running time is $\tilde{O}(\frac{n}{b} + \frac{n}{\sqrt{g}} + \frac{bgn}{t} + \frac{gn^{3/2+\delta}}{tz^{1/2}})$, and we obtain an $(O(1), \tilde{O}(\frac{n}{b} + \frac{n}{\sqrt{g}}))$-approximation. (The expected bound can be converted to worst-case by placing a time limit and re-running logarithmically many times.) Choosing $g = b^2$ yields the following result:

▶ **Lemma 9.** *Given parameters b and $z \in [1, n]$ and any constant $\varepsilon > 0$, there exists a data structure for the dynamic set cover problem for $O(n)$ upper halfspaces and $O(n)$ points in 3D that supports insertions and deletions in $\tilde{O}(z)$ amortized time and can find the value of an $(O(1), \tilde{O}(\frac{n}{b}))$-approximation w.h.p. in $\tilde{O}(\frac{n}{b} + \frac{b^3 n}{\mathrm{OPT}} + \frac{b^2 n^{3/2+\delta}}{\mathrm{OPT} \cdot z^{1/2}})$ time for any constant $\delta > 0$.*

A minor technicality is that when applying Lemma 8, we have assumed that the optimal value is finite. The problem of checking whether a solution exists, i.e., whether a point set is covered by a set of halfspaces (or more generally, maintaining the lowest-depth point), subject to insertions and deletions of points and halfspaces, has already been solved before by Chan [8, Theorem 4.1], who gave a fully dynamic algorithm with $\tilde{O}(n^{2/3})$ time per operation (based on augmenting partition trees with counters, similar to what we have done here).

Combining the algorithms. Finally, we combine all three algorithms:

1. When $\mathrm{OPT} \le n^{2/9}$, we use the algorithm for small OPT; the running time is $\tilde{O}(\mathrm{OPT} \cdot n^{2/3+\delta}) \le \tilde{O}(n^{8/9+\delta})$.
2. When $n^{2/9} < \mathrm{OPT} \le n^{10/13}$, we use the algorithm for medium OPT; the running time is $\tilde{O}(\frac{n^{7/8}}{\mathrm{OPT}^{1/8}} + \frac{n}{\sqrt{\mathrm{OPT}}} + \mathrm{OPT}^{1/3} n^{2/3+O(\delta)}) \le O(n^{12/13+O(\delta)})$.
3. When $\mathrm{OPT} > n^{10/13}$, we use the algorithm for large OPT with $b = \tilde{\Theta}(n^{3/13})$ and $z = n^{7/13}$, so that an $(O(1), \tilde{O}(\frac{n}{b}))$-approximation is indeed an $O(1)$-approximation; the running time is $\tilde{O}(\frac{n}{b} + \frac{b^3 n}{\mathrm{OPT}} + \frac{b^2 n^{3/2+\delta}}{\mathrm{OPT} \cdot z^{1/2}}) \le O(n^{12/13+O(\delta)})$.

▶ **Theorem 10.** *There exists a data structure for the dynamic set cover problem for $O(n)$ upper halfspaces and $O(n)$ points in 3D that maintains the value of an $O(1)$-approximate solution w.h.p. with $O(n^{12/13+\delta})$ amortized insertion and deletion time for any constant $\delta > 0$.*

References

1 Anna Adamaszek, Sariel Har-Peled, and Andreas Wiese. Approximation schemes for independent set and sparse subsets of polygons. *Journal of the ACM*, 66(4):29:1–29:40, 2019. doi:10.1145/3326122.

2 Pankaj K. Agarwal, Hsien-Chih Chang, Subhash Suri, Allen Xiao, and Jie Xue. Dynamic geometric set cover and hitting set. In *Proceedings of the 36th Symposium on Computational Geometry (SoCG)*, volume 164, pages 2:1–2:15, 2020. doi:10.4230/LIPIcs.SoCG.2020.2.

3 Pankaj K. Agarwal and Jeff Erickson. Geometric range searching and its relatives. In B. Chazelle, J. E. Goodman, and R. Pollack, editors, *Advances in Discrete and Computational Geometry*, pages 1–56. AMS Press, 1999. URL: `http://jeffe.cs.illinois.edu/pubs/survey.html`.

4 Pankaj K. Agarwal and Jiangwei Pan. Near-linear algorithms for geometric hitting sets and set covers. *Discrete & Computational Geometry*, 63(2):460–482, 2020. Preliminary version in SoCG'14. `doi:10.1007/s00454-019-00099-6`.

5 Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, and Angela Y. Wu. An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. *Journal of the ACM*, 45(6):891–923, 1998. `doi:10.1145/293347.293348`.

6 Hervé Brönnimann and Michael T. Goodrich. Almost optimal set covers in finite VC-dimension. *Discrete & Computational Geometry*, 14(4):463–479, 1995.

7 Timothy M. Chan. Random sampling, halfspace range reporting, and construction of $(\leq k)$-levels in three dimensions. *SIAM Journal on Computing*, 30(2):561–575, 2000. `doi:10.1137/S0097539798349188`.

8 Timothy M. Chan. Semi-online maintenance of geometric optima and measures. *SIAM Journal on Computing*, 32(3):700–716, 2003. `doi:10.1137/S0097539702404389`.

9 Timothy M. Chan. A dynamic data structure for 3-d convex hulls and 2-d nearest neighbor queries. *Journal of the ACM*, 57(3):16:1–16:15, 2010. `doi:10.1145/1706591.1706596`.

10 Timothy M. Chan. Optimal partition trees. *Discrete & Computational Geometry*, 47(4):661–690, 2012. `doi:10.1007/s00454-012-9410-z`.

11 Timothy M. Chan and Qizheng He. Faster approximation algorithms for geometric set cover. In *Proceedings of the 36th Symposium on Computational Geometry (SoCG)*, volume 164, pages 27:1–27:14, 2020. `doi:10.4230/LIPIcs.SoCG.2020.27`.

12 Timothy M. Chan and Konstantinos Tsakalidis. Optimal deterministic algorithms for 2-d and 3-d shallow cuttings. *Discrete & Computational Geometry*, 56(4):866–881, 2016.

13 Kenneth L. Clarkson. New applications of random sampling in computational geometry. *Discrete & Computational Geometry*, 2:195–222, 1987. `doi:10.1007/BF02187879`.

14 Kenneth L. Clarkson. Algorithms for polytope covering and approximation. In *Workshop on Algorithms and Data Structures*, pages 246–252, 1993.

15 Kenneth L. Clarkson and Kasturi Varadarajan. Improved approximation algorithms for geometric set cover. *Discrete & Computational Geometry*, 37(1):43–58, 2007.

16 Artur Czumaj, Funda Ergün, Lance Fortnow, Avner Magen, Ilan Newman, Ronitt Rubinfeld, and Christian Sohler. Approximating the weight of the Euclidean minimum spanning tree in sublinear time. *SIAM Journal on Computing*, 35(1):91–109, 2005. `doi:10.1137/S0097539703435297`.

17 Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Mark H. Overmars. *Computational Geometry: Algorithms and Applications*. Springer, 3rd edition, 2008. URL: `https://www.worldcat.org/oclc/227584184`.

18 Greg N. Frederickson. Fast algorithms for shortest paths in planar graphs, with applications. *SIAM Journal on Computing*, 16(6):1004–1022, 1987.

19 Monika Henzinger, Stefan Neumann, and Andreas Wiese. Dynamic approximate maximum independent set of intervals, hypercubes and hyperrectangles. In *Proceedings of the 36th Symposium on Computational Geometry (SoCG)*, volume 164, pages 51:1–51:14, 2020. `doi:10.4230/LIPIcs.SoCG.2020.51`.

20 Richard J. Lipton and Robert Endre Tarjan. Applications of a planar separator theorem. *SIAM Journal on Computing*, 9(3):615–627, 1980. `doi:10.1137/0209046`.

21 Chih-Hung Liu, Evanthia Papadopoulou, and D. T. Lee. An output-sensitive approach for the L_1/L_∞ k-nearest-neighbor Voronoi diagram. In *Proceedings of the 19th Annual European Symposium on Algorithms (ESA)*, volume 6942 of *Lecture Notes in Computer Science*, pages 70–81. Springer, 2011. `doi:10.1007/978-3-642-23719-5_7`.

22 Jiří Matoušek. Efficient partition trees. *Discrete & Computational Geometry*, 8(3):315–334, 1992.

23 Jiří Matoušek. Reporting points in halfspaces. *Computational Geometry*, 2(3):169–186, 1992.

24 Jiří Matoušek. Range searching with efficient hierarchical cutting. *Discrete & Computational Geometry*, 10:157–182, 1993. doi:10.1007/BF02573972.

25 Nabil H. Mustafa, Rajiv Raman, and Saurabh Ray. Quasi-polynomial time approximation scheme for weighted geometric set cover on pseudodisks and halfspaces. *SIAM Journal on Computing*, 44(6):1650–1669, 2015. doi:10.1137/14099317X.

26 Nabil H. Mustafa and Saurabh Ray. Improved results on geometric hitting set problems. *Discrete & Computational Geometry*, 44(4):883–895, 2010. doi:10.1007/s00454-010-9285-9.

27 Micha Sharir. On k-sets in arrangement of curves and surfaces. *Discrete & Computational Geometry*, 6:593–613, 1991. doi:10.1007/BF02574706.

Approximating the (Continuous) Fréchet Distance

Connor Colombe
The University of Texas at Austin, TX, USA

Kyle Fox
The University of Texas at Dallas, Richardson, TX, USA

Abstract

We describe the first strongly subquadratic time algorithm with subexponential approximation ratio for approximately computing the Fréchet distance between two polygonal chains. Specifically, let P and Q be two polygonal chains with n vertices in d-dimensional Euclidean space, and let $\alpha \in [\sqrt{n}, n]$. Our algorithm deterministically finds an $O(\alpha)$-approximate Fréchet correspondence in time $O((n^3/\alpha^2)\log n)$. In particular, we get an $O(n)$-approximation in near-linear $O(n \log n)$ time, a vast improvement over the previously best know result, a linear time $2^{O(n)}$-approximation. As part of our algorithm, we also describe how to turn any approximate decision procedure for the Fréchet distance into an approximate optimization algorithm whose approximation ratio is the same up to arbitrarily small constant factors. The transformation into an approximate optimization algorithm increases the running time of the decision procedure by only an $O(\log n)$ factor.

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Approximation algorithms analysis

Keywords and phrases Fréchet distance, approximation algorithm, approximate decision procedure

Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.26

Related Version *Full Version*: `https://arxiv.org/abs/2007.07994`

Acknowledgements Most of this work was done while the first author was a student at the University of Texas at Dallas. The authors would like to thank Karl Bringmann and Marvin Künnemann for some helpful discussions concerning turning an approximate decision procedure into a proper approximation algorithm.

1 Introduction

The Fréchet distance is a commonly used method of measuring the similarity between a pair of curves. Both its standard (continuous) and discrete variants have seen use in map construction and mapping [5, 16], handwriting recognition [27], and protein alignment [23].

Formally, it is defined as follows: Let $P : [1, m] \to \mathbb{R}^d$ and $Q : [1, n] \to \mathbb{R}^d$ be two curves in d-dimensional Euclidean space. We'll assume P and Q are represented as **polygonal chains**, meaning there exist ordered **vertex** sequences $\langle p_1, \ldots, p_m \rangle$ and $\langle q_1, \ldots, q_n \rangle$ such that $P(i) = p_i$ for all $1 \leq i \leq m$, $Q(j) = q_j$ for all $1 \leq j \leq n$, and both P and Q are linearly parameterized along line segments or **edges** between these positions. We define a **re-parameterization** $\sigma : [0, 1] \to [1, m]$ of P as any continuous, non-decreasing function such that $\sigma(0) = 1$ and $\sigma(1) = m$.[1] We define a re-parameterization $\theta : [0, 1] \to [1, n]$ of Q similarly. We define a **Fréchet correspondence** between P and Q as a pair (σ, θ) of re-parameterizations of P and Q respectively, and we say any pair of reals $(\sigma(r), \theta(r))$ for any $0 \leq r \leq 1$ are **matched** by the correspondence. Let $\mathtt{d}(p, q)$ denote the Euclidean distance between points p and q in $\mathbb{R}^d$. The **cost** of the correspondence is defined as

$$\mu((\sigma, \theta)) := \max_{0 \leq r \leq 1} \mathtt{d}(P(\sigma(r)), Q(\theta(r))).$$

[1] Re-parameterizations are normally required to be bijective, but we relax this requirement to simplify definitions and arguments throughout the paper.

Let Π_{FD} denote the set of all Fréchet correspondences between P and Q. The **(continuous) Fréchet distance** of P and Q is defined as

$$\mathsf{FD}(P,Q) := \min_{(\sigma,\theta)\in\Pi_{\mathsf{FD}}} \mu((\sigma,\theta)).$$

The standard intuition given for this definition is to imagine a person and their dog walking along P and Q, respectively, without backtracking. The person must keep the dog on a leash, and the goal is to pace their walks as to minimize the length of leash needed to keep them connected. There also exists a variant of the distance called the **discrete Fréchet distance** where the input consists of two finite *point sequences*. Here, we replace the person and dog by two frogs. Starting with both frogs on the first point of their sequences, we must iteratively move the first, the second, or both frogs to the next point in their sequences. As before, the goal is to minimize the maximum distance between the frogs.

Throughout this paper, we assume $2 \leq m \leq n$. We can easily compute the *discrete* Fréchet distance in $O(mn)$ time using dynamic programming. The first polynomial time algorithm for computing the continuous case was described by Alt and Godau [6]. They use parametric search [17, 25] and a quadratic time decision procedure (see Section 2) to compute the Fréchet distance in $O(mn \log n)$ time. Almost two decades passed before Agarwal et al. [3] improved the running time for the discrete case to $O(mn \log\log n/\log n)$. Buchin et al. [14] later improved the running time for the continuous case to $O(mn(\log\log n)^2)$ (these latter two results assume we are working in the word RAM model of computation).

Recently, Gudmundsson et al. [21] described an $O(n \log n)$ time algorithm for computing the continuous distance between chains P and Q assuming all edges have length a sufficiently large constant larger than $\mathsf{FD}(P,Q)$. In short, having long edges allows one to greedily move the person and dog along their respective chains while keeping their leash length optimal.

From this brief history, one may assume substantially faster algorithms are finally forthcoming for general cases of the continuous and discrete Fréchet distance. Unfortunately, more meaningful improvements may not be possible; Bringmann [10] showed that **strongly subquadratic** ($n^{2-\Omega(1)}$) time algorithms would violate the *Strong Exponential Time Hypothesis* (SETH) that solving CNF-SAT over n variables requires $2^{(1-o(1))n}$ time [22].

Therefore, we are motivated to look for fast *approximation algorithms* for these problems. Aronov et al. [8] described a $(1+\varepsilon)$-approximation algorithm for the discrete Fréchet distance. This algorithm runs in subquadratic and often near-linear time if P or Q fall into one of a few different "realistic" families of curves such as ones modeling protein backbones. Driemel et al. [18] describe a $(1+\varepsilon)$-approximation for the continuous Fréchet distance that again runs more quickly if one of the curves belongs to a realistic family than it would otherwise. This latter algorithm was improved for some cases by Bringmann and Künnemann [12]. In the same work mentioned above, Gudmundsson et al. [21] described a $\sqrt{d}$-approximation algorithm that runs in linear time if the input polygonal chains have sufficiently long edges.

Approximation appears more difficult when the input is arbitrary. Bringmann [10] showed there is no strongly subquadratic time 1.001-approximation for the Fréchet distance, assuming SETH. For arbitrary point sequences, Bringmann and Mulzer [13] described an $O(\alpha)$-approximation algorithm for the discrete distance for any $\alpha \in [1, n/\log n]$ that runs in $O(n \log n + n^2/\alpha)$ time. Chan and Rahmati [15] later described an $O(n \log n + n^2/\alpha^2)$ time $O(\alpha)$-approximation algorithm for the discrete distance for any $\alpha \in [1, \sqrt{n/\log n}]$.

For the *continuous* Fréchet distance over arbitrary polygonal chains, the only strongly subquadratic time algorithm known with bounded approximation ratio runs in linear time but has an *exponential* worst case approximation ratio of $2^{\Theta(n)}$. This result is described in the same paper of Bringmann and Mulzer [13] mentioned above. We note that there is also

a substantial body of work on the (approximate) nearest neighbor problem using Fréchet distance as the metric; see Mirzanezhad [26] for a survey of recent results. These results assume the query curve or the curves being searched are short, so they do not appear directly useful in approximating the Fréchet distance between two curves of arbitrary complexity.

The closely related problems of computing the dynamic time warping and geometric edit distances have a similar history to that of the discrete Fréchet distance.[2] They have straightforward quadratic time dynamic programming algorithms that have been improved by (sub-)polylogarithmic factors for some low dimensional cases [20]; substantial improvements to these algorithms violate SETH or other complexity theoretic assumptions [9, 1, 11, 2]; and there are fast $(1+\varepsilon)$-approximation algorithms specialized for realistic input sequences [4, 28]. And, there exist some approximation results for arbitrary point sequences as well. Kuszmaul [24] described $O((n^2/\alpha)\operatorname{polylog} n)$ time $O(\alpha)$-approximation algorithms for dynamic time warping distance over point sequences in *well separated tree metrics* of exponential spread and geometric edit distance over point sequences in arbitrary metrics. Fox and Li [19] described a randomized $O(n\log^2 n + (n^2/\alpha^2)\log n)$ time $O(\alpha)$-approximation algorithm for geometric edit distance for points in low dimensional Euclidean space. Even better approximation algorithms exist for the traditional string edit distance where all substitutions have cost exactly 1; see, for example, Andoni and Nosatzki [7].

Each of the above problems for point *sequences* admit strongly subquadratic approximation algorithms with polynomial approximation ratios when the input comes from low dimensional Euclidean space. However, such a result remains conspicuously absent for the continuous Fréchet distance over arbitrary polygonal chains. One may naturally assume results for the discrete Fréchet distance extend to the continuous case. However, one advantage of discrete Fréchet distance over the continuous case is that input points can only be matched with other input points. The fact that vertices can match with edge interiors in the continuous case makes it much more difficult to make approximately optimal decisions. In addition, we can no longer depend upon certain data structures for testing equality of subsequences in constant time. These structures are largely responsible for the relatively small running times seen in the algorithms of Chan and Rahmati [15] and Fox and Li [19].

Our results

We describe the first strongly subquadratic time algorithm with subexponential approximation ratio for computing Fréchet correspondences between polygonal chains. Let P and Q be two polygonal chains of m and n vertices, respectively, in d-dimensional Euclidean space, and let $\alpha \in [\sqrt{n}, n]$. Again, we assume $m \le n$. Our algorithm deterministically finds a Fréchet correspondence between P and Q of cost $O(\alpha)\cdot\mathsf{FD}(P,Q)$ in time $O((n^3/\alpha^2)\log n)$. In particular, we get an $O(n)$-approximation in near-linear $O(n\log n)$ time, a vast improvement over Bringmann and Mulzer's [13] linear time $2^{O(n)}$-approximation for continuous Fréchet distance. Our algorithm employs a novel combination of ideas from the original exact algorithm of Alt and Godau [6] for continuous Fréchet distance, the algorithm of Chan and Rahmati [15] for approximating the discrete Fréchet distance, and Gudmundsson et al.'s [21] greedy approach for computing the Fréchet distance between chains with long edges.

[2] The dynamic time warping distance is defined similarly to the discrete Fréchet distance, except the goal is to minimize the *sum* of distances between the frogs over all pairs of points they stand upon. The geometric edit distance can be defined as the minimum number of point insertions and deletions plus the minimum total cost of point substitutions needed to transform one input sequence into another. The cost of a substitution is the distance between its points.

Let $\delta > 0$. We describe an **approximate decision procedure** that either determines $\mathsf{FD}(P, Q) > \delta$ or finds a Fréchet correspondence of cost $O(\alpha) \cdot \delta$. The *exact* decision procedure of Alt and Godau [6] computes a set of *reachability intervals* in the *free space diagram* of P and Q with respect to δ (see Section 2). These intervals represent all points on a single edge of Q that can be matched to a vertex of P (or vice versa) in a Fréchet correspondence of cost at most δ. For our approximate decision procedure, we compute a set of *approximate reachability intervals* such that the re-parameterizations realizing these intervals have cost $O(\alpha) \cdot \delta$. We cannot afford to compute intervals for all $\Theta(mn)$ vertex-edge pairs, so we instead focus on $O(n^2/\alpha^2)$ vertex-edge pairs as described below that contain the first and last vertices and edges of both chains. The approximate interval we compute for any vertex-edge pair contains the exact interval for that same pair. So if $\mathsf{FD}(P, Q) \leq \delta$, we are guaranteed (p_m, q_n) is approximately reachable and our desired Fréchet correspondence exists.

The vertex-edge pairs chosen to hold the approximate reachability intervals follow from ideas of Chan and Rahmati [15]. Similar to them, we place a grid of side length $\alpha \cdot \delta$ so that at most $O(n/\alpha)$ vertices of P and Q lie within distance 3δ of the side of a grid box. We call these $O(n/\alpha)$ vertices *bad* and the rest *good*. Also, we call any edge with a bad endpoint bad. Our approximate reachability intervals involve only bad edges and vertices with at least one bad incident edge. To compute these intervals, we describe a method for tracing how a Fréchet correspondence of cost δ must behave starting from one approximate reachability interval until it reaches some others we wish to compute. Recall, an approximate reachability interval corresponds to pairs of points on P and Q that could be matched together. Either the next edge of P or Q after one of these pairs to leave a box is good and therefore long, or it is bad, and we can afford to compute some new approximate reachability intervals using this edge. We can easily compute correspondences between long edges and arbitrary length edges on the other curve, and we can greedily match the portions of the curves before they leave the box at cost at most $O(\alpha) \cdot \delta$. The traces take only $O(n)$ time each, and we perform at most $O(n^2/\alpha^2)$ traces, so our decision procedure takes $O(n^3/\alpha^2)$ time total.

We would like to use our approximate decision procedure as a black box to compute a Fréchet correspondence of cost $O(\alpha) \cdot \mathsf{FD}(P, Q)$ without knowing $\mathsf{FD}(P, Q)$ in advance. Unfortunately, we are unaware of any known general method to do so.[3] Therefore, we describe how to turn any approximate decision procedure into an algorithm with the same approximation ratio up to arbitrarily small constant factors after an $O(\log n)$ factor increase in running time. In particular, any improvement to our approximate decision procedure would immediately carry over to our overall approximation algorithm. Our method involves binary searching over a set of $O(n)$ values approximating distances between pairs of vertices. If there is a large gap between the Fréchet distance and the nearest of these values, we can *simplify* both P and Q without losing much accuracy in the Fréchet distance computation while allowing us to use the long edge exact algorithm of Gudmundsson et al. [21].

The rest of our paper is organized as follows. We describe preliminary notions in Section 2. We describe our decision procedure in Section 3 and how to turn it into an approximation algorithm in Section 4. We conclude with some closing thoughts in Section 5.

[3] Bringmann and Künnemann [12, Lemma 2.1] claim there exists a general method for turning an approximate decision procedure into an approximate optimization algorithm when the approximation ratio of the decision procedure is at most 2. However, they rely on a method of Driemel et al. [18] that uses certain structural properties of the input polygonal chains that we cannot assume.

2 Preliminaries

Let $R : [1, n] \to \mathbb{R}^d$ be a polygonal chain in d-dimensional Euclidean space. We let $R[r, r']$ denote the restriction of R to $[r, r']$. In other words, the notation refers to the portion of R between points $R(r)$ and $R(r')$. We generally use s to refer to members of the domain of a polygonal chain P and t to refer to members of the domain of a polygonal chain Q. We use i and j, respectively, when these members are integers. Recall, $p_i = P(i)$ for all $1 \leq i \leq m$ and $q_j = Q(j)$ for all $1 \leq j \leq n$. We use superscript notation (s^a) to label particular members of these domains (and *not* to take the ath power of s), and we use subscript notation (s_k) when we are working with an ordered list of these members.

Free space diagram and reachability

Let $P : [1, m] \to \mathbb{R}^d$ and $Q : [1, n] \to \mathbb{R}^d$ be polygonal chains. Fix some $\delta > 0$. Alt and Godau [6] introduced the **free space diagram** to decide if $\mathsf{FD}(P, Q) \leq \delta$. It consists of a set of pairs $F = \{(s, t) \in [1, m] \times [1, n]\}$. Each $(s, t) \in F$ represents the pair of points $P(s)$ and $Q(t)$. Point $(s, t) \in F$ is **free** if $\mathtt{d}(P(s), Q(t)) \leq \delta$. The **free space** $\mathcal{D}_{\leq\delta}(P, Q)$ consists of all free points between P and Q for a given δ. Formally, it is given by the set $\mathcal{D}_{\leq\delta}(P, Q) := \{(s, t) \in [1, m] \times [1, n] : \mathtt{d}(P(s), Q(t)) \leq \delta\}$. We say that a point $(s', t') \in F$ is **reachable** if there exists an s and t-monotone path from $(1, 1)$ to (s', t') through $\mathcal{D}_{\leq\delta}(P, Q)$.

The standard procedure for determining if $\mathsf{FD}(P, Q) \leq \delta$ divides F into cells $C_{i,j} := [i - 1, i] \times [j - 1, j]$ for all $i \in \langle 2, \ldots, m\rangle$ and $j \in \langle 2, \ldots, n\rangle$. The intersection of a cell $C_{i,j}$ with the free space is convex [6]. The intersection of an edge of the free space diagram cell $C_{i,j}$ with the free space forms a **free space interval**. The subset of reachable points within a free space interval form what is called an (exact) **reachability interval**. We say a Fréchet correspondence (σ, θ) between P and Q **uses** or **passes through** a reachability interval if there exists some point $(\sigma(r), \theta(r))$ within that interval.

Given the bottom and left reachability intervals of a free space diagram cell, we can compute the top and right reachability intervals of the same cell in $O(1)$ time [6]. The exact decision procedure loops through the cells in increasing order of i and j, computing reachability intervals one-by-one. Let $\alpha \in [\sqrt{n}, n]$. We cannot afford to compute all $\Theta(mn)$ reachability intervals, so instead we compute $O(n^2/\alpha^2)$ **(α)-approximate reachability intervals**. The approximate reachability intervals are subsets of the free space intervals such that for any point (s, t) on an approximate reachability interval, there exists a Fréchet correspondence between $P[1, s]$ and $Q[1, t]$ of cost $O(\alpha) \cdot \delta$. We express exact or approximate reachability intervals by the subset of F they contain; for example, given $j - 1 \leq t^a \leq t^b \leq j$, we will use $\{i\} \times [t^a, t^b]$ to refer to an interval on the right side of cell $C_{i,j}$.

Grids, good points, bad points, and dangerous points

Chan and Rahmati [15] utilize a d-dimensional grid to create the useful notion of good and bad vertices for their discrete Fréchet distance approximation algorithm. We adopt their use of a d-dimensional grid. Unlike Chan and Rahmati, however, we are no longer working with sequences of discrete points but instead polygonal chains. We must therefore define new constructs of good and bad that work better for our problem's input.

Let $P : [1, m] \to \mathbb{R}^d$ and $Q : [1, n] \to \mathbb{R}^d$ be two polygonal chains. Fix $\delta > 0$ and $\alpha \in [\sqrt{n}, n]$. Let G be a d-dimensional grid consisting of **boxes** of side length $\alpha \cdot \delta$. (We do not use the term *cell* here to avoid confusion with the free space diagram.) We say a vertex of P or Q is **good** if it is more than distance 3δ from any edge of G. If a vertex is not good, then we call it **bad**. For simplicity, we also designate p_1, q_1, p_m, and q_n as bad, regardless of their position within boxes of G.

We also extend the constructs of good and bad to the edges of P and Q. We say an edge on either chain is **good** if both its endpoints are good vertices. Otherwise, the edge is **bad**. Lastly, we say that a vertex is **dangerous** (but not necessarily good or bad) if at least one of its incident edges is bad. Chan and Rahmati [15, Lemma 1] demonstrate how to compute a grid G with $O(n/\alpha)$ bad vertices in $O(n)$ time. Because each bad vertex has up to two incident edges, there are also $O(n/\alpha)$ bad edges. Each bad edge is incident to two vertices, so there are $O(n/\alpha)$ dangerous vertices as well. Our approximate decision procedure will compute approximate reachability intervals only between dangerous vertices and bad edges. Therefore, there will be at most $O(n^2/\alpha^2)$ such intervals.

Curve simplification

Let $R : [1, n] \to \mathbb{R}^d$ be a polygonal chain with vertices $\langle r_1, \ldots, r_n \rangle$. Our approximation algorithm relies on a method for simplifying chains so their edges are not too short. We slightly modify of a procedure of Driemel et al. [18]. Let $\nu > 0$ be a parameter. We mark r_1 and set it as the *current vertex*. We then repeat the following procedure until we no longer have a designated current vertex. We scan R from the current vertex until reaching the first vertex r_i of distance at least ν from the current vertex. We mark r_i, set it as the current vertex, and perform the next iteration of the loop. The **ν-simplification** of R, denoted $\hat{R}$, is the polygonal chain consisting of exactly the marked vertices in order. Note that unlike Driemel et al. [18], we do not require the final vertex of R to be marked. We can easily verify that all edges of $\hat{R}$ have length at least ν. Also, $\mathsf{FD}(R, \hat{R}) \leq \nu$ [18, Lemma 2.3].

3 Approximate decision procedure

In this section, we present our $O(\alpha)$-approximate decision procedure. Let $P : [1, m] \to \mathbb{R}^d$ and $Q : [1, n] \to \mathbb{R}^d$ be two polygonal chains in d-dimensional Euclidean space as defined before, and let $\alpha \in [\sqrt{n}, n]$. Let $\delta > 0$. We begin by computing the grid G along with $O(n/\alpha)$ bad edges and points as defined in Section 2. We then explicitly compute and record a set of $O(n^2/\alpha^2)$ approximate reachability intervals between dangerous vertices and bad edges. To compute these intervals, we occasionally perform a linear time greedy search for a good correspondence. We describe this greedy search procedure in Section 3.1 before giving the remaining details of the decision procedure in Section 3.2.

3.1 Greedy mapping subroutines

We describe a pair of subroutines for greedily computing Fréchet correspondences along lengths of P and Q. The first of these procedures GreedyMappingP(i, t) takes as its input an integer $i \in \langle 1, \ldots, m \rangle$ such that p_i is a good vertex of P along with a real value $t \in [1, n]$ such that $\mathsf{d}(p_i, Q(t)) \leq \delta$. Informally, the procedure does the following: Suppose there exists a Fréchet correspondence (σ, θ) between P and Q of cost at most δ that maps p_i "close to" $Q(t)$. Procedure GreedyMappingP(i, t) essentially follows P and Q from box to box, discovering groups of points that must be matched by (σ, θ). When there is too much ambiguity in what must be matched to continue searching greedily, it outputs a set of approximate reachability intervals, including one used by (σ, θ). While we can infer which boxes pairs of matched points belong to, it may still be unclear exactly which pairs appear in (σ, θ). Also, the procedure may output intervals despite (σ, θ) not existing in the first place! Therefore, we can only guarantee the intervals can be reached using a correspondence of cost at most $O(\alpha) \cdot \delta$. We define another procedure GreedyMappingQ(j, s) similarly, exchanging the roles of P and Q. As they are rather technical, the precise definitions of these procedures are best expressed in the following lemmas.

▶ **Lemma 1.** *Let $i \in \langle 1, \ldots, m \rangle$ and $t \in [1, n]$ such that p_i is good and $\mathtt{d}(p_i, Q(t)) \leq \delta$. Procedure* GreedyMappingP(i, t) *outputs zero or more approximate reachability intervals between a bad edge of P or Q and a dangerous vertex of Q or P, respectively. For each pair $(s', t') \in [i, m] \times [t, n]$ in an approximate reachability interval computed by the procedure, there exists a Fréchet correspondence of cost $O(\alpha) \cdot \delta$ between $P[i, s']$ and $Q[t, t']$. Procedure* GreedyMappingQ(j, s) *has the same properties with the roles of P and Q exchanged.*

▶ **Lemma 2.** *Let $i \in \langle 1, \ldots, m \rangle$ and $t \in [1, n]$ such that p_i is good and $\mathtt{d}(p_i, Q(t)) \leq \delta$. Suppose there exists a Fréchet correspondence (σ, θ) between P and Q of cost at most δ that matches i with some $t^* \geq t$ such that every point of $Q[t, t^*]$ is at most distance 3δ from p_i. Then, (σ, θ) passes through at least one approximate reachability interval output by procedure* GreedyMappingP(i, t). *Procedure* GreedyMappingQ(j, s) *has the same properties with the roles of P and Q exchanged.*

We now describe GreedyMappingP(i, t). Procedure GreedyMappingQ(j, s) has an analogous description, with the roles of P and Q exchanged. We provide details on the implementation of GreedyMappingP(i, t) along with intuition for the steps it uses. We defer formal proofs of the above lemmas until we are done describing the procedure. The symmetric roles of P and Q in the problem input lead to a fair bit of redundancy in the description of GreedyMappingP(i, t). Accordingly, we defer to symmetry a few times in the description below and give full details of the procedure in the full version of the paper.

To begin, observe p_i and $Q(t)$ lie in the same box B of grid G, because p_i is good and $\mathtt{d}(p_i, Q(t)) \leq \delta$. We first follow P and Q to see where they leave B: Let $s^e = m$ if P never leaves B after p_i. Otherwise, let s^e be the minimum value in $(i, m]$ such that $P(s^e)$ lies on the boundary of B (the 'e' stands for *exit*). Define t^e similarly for Q. See Figure 1.

If either curve ends before leaving B, then the rest of the other curve needs to stay near B if a correspondence like in Lemma 2 exists. Therefore, if $s^e = m$ (resp. $t^e = n$), we check if all points of $Q[t, n]$ (resp. $P[i, m]$) lie in or within distance δ of B. If so, we output the trivial approximate reachability interval of $\{(m, n)\}$ and terminate the procedure. Otherwise, we output zero approximate reachability intervals.

From here on, we assume $s^e \neq m$ and $t^e \neq n$. Let $i^e \in \langle 1, \ldots, m \rangle$ such that $i^e - 1 \leq s^e \leq i^e$, and define j^e similarly. We begin by considering cases where a curve leaves box B along a good edge. Here, a correspondence as described in Lemma 2 must match a portion of the other curve to the good edge. Fortunately, we can guess approximately where that portion of the other curve begins and ends. Afterward, the other endpoint of the good edge serves as a suitable parameter for a recursive call to one of our greedy mapping procedures.

Specifically, suppose edge $P[i^e - 1, i^e]$ is good. In this case, let t^f be the minimum value in $(t, n]$ such that $\mathtt{d}(p_{i^e}, Q(t^f)) \leq \delta$, and let t^c be the *maximum* value in $[t, t^f)$ such that $\mathtt{d}(p_{i^e - 1}, Q(t^c)) \leq \delta$ (the 'f' stands for *far*, and the 'c' stands for *close*). See Figure 2. We check if every point of $Q[t, t^c]$ lies in or within distance δ of B and if $\mathsf{FD}(P[i^e - 1, i^e], Q[t^c, t^f]) \leq \delta$. If so, we run GreedyMappingP(i^e, t^f) and use its output. Otherwise, we output zero approximate reachability intervals. The case where $P[i^e - 1, i^e]$ is bad but $Q[j^e - 1, j^e]$ is good is handled symmetrically. See the full version of the paper.

From here on, we assume neither curve leaves box B through a good edge. Suppose there is a correspondence (σ, θ) as described in Lemma 2. Further suppose the reparameterized walks along P and Q leave box B along P before Q. In this case, we can show that $P(s^e)$ is matched with a point on a bad edge of Q. Accordingly, we iterate over the bad edges of Q that appear before Q leaves box B, computing sufficiently large approximate reachability intervals along the top and right sides of free space diagram cells for $P[i^e - 1, i^e]$ and those bad edges of Q. Both of the edges for each of these cells are bad, so the intervals we compute are between bad edges and dangerous vertices.

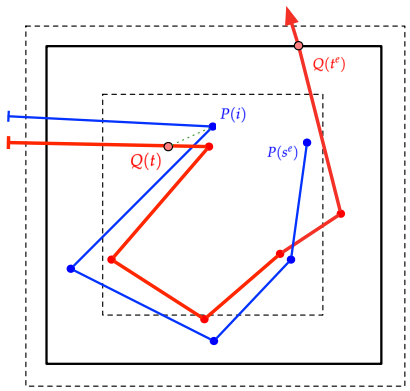


Figure 1 Basic setup for GreedyMappingP(i, t).

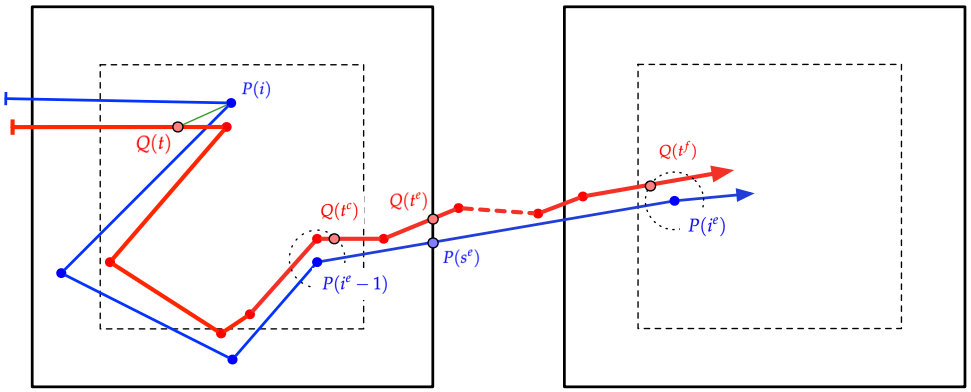


Figure 2 GreedyMappingP(i, t): The case where $P[i^e - 1, i^e]$ is good.

Specifically, let $t \le t_1 < t_2 < \cdots < t_\ell \le t^e$ be the list of first positions along their respective edges of Q such that $\mathtt{d}(P(s^e), Q(t_k)) \le \delta$ for each $k \in \langle 1, \ldots, \ell \rangle$. See Figure 3, left. Observe that each edge containing a point t_k must be bad, because Q does not leave B along a good edge, and no good edge with two endpoints in B lies within distance δ of $P(s^e)$. For each $k \in \langle 1, \ldots, \ell \rangle$, we do the following: Let $j_k \in \langle 1, \ldots, n \rangle$ such that $j_k - 1 \le t_k \le j_k$. Let t_k^a be the minimum value in $[t_k, j_k]$ such that $\mathtt{d}(p_{i^e}, Q(t_k^a)) \le \delta$ and let t_k^b be the maximum value in $[t_k, j_k]$ such that $\mathtt{d}(p_{i^e}, Q(t_k^b)) \le \delta$. If t_k^a and t_k^b are well-defined, then we designate the interval $\{i^e\} \times [t_k^a, t_k^b]$ as approximately reachable. (If we have already designated a subset of $\{i^e\} \times [j_k - 1, j_k]$ as approximately reachable earlier in the decision procedure, then we extend the approximately reachable area by taking the union with the old interval. Every interval of $\{i^e\} \times [j_k - 1, j_k]$ we compute will end at (i^e, t_k^b), so the union is also an interval.) Similarly, let s_k^a be the minimum value in $[s^e, i^e]$ such that $\mathtt{d}(P(s_k^a), q_{j_k}) \le \delta$ and let s_k^b be the maximum value in $[s^e, i^e]$ such that $\mathtt{d}(P(s_k^b), q_{j_k}) \le \delta$. If s_k^a and s_k^b are well-defined, then we designate the interval $[s_k^a, s_k^b] \times \{j_k\}$ as approximately reachable. See Figure 3, right.

It is also possible that a good correspondence has the walk along Q leave box B first. So, *in addition* to the above set of approximate reachability intervals, we create some based on points of P between p_i and $P(s^e)$ that are near $Q(t^e)$. See the full version of the paper.

We have concluded our description of GreedyMappingP(i, t). While it is still non-trivial, the proof of Lemma 1 follows relatively easily from the description of GreedyMappingP(i, t). The details appear in the full version of the paper. The proof of Lemma 2 is more surprising and interesting, so it appears below.

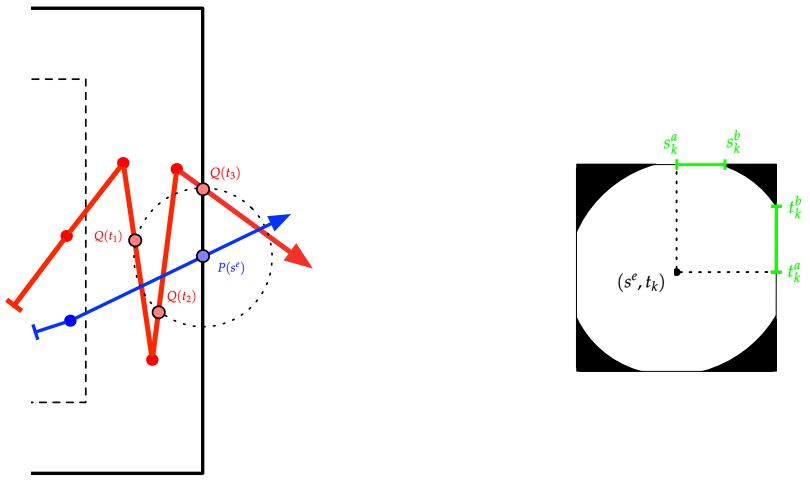


Figure 3 Left: Definition of $\langle t_1, \ldots, t_\ell \rangle$. Right: Designating approximate reachability intervals near a pair (s^e, t_k). The clipped ellipse coincides with the free points inside cell C_{i^e,j_k}.

Proof of Lemma 2. We use the same notation as given in the description of GreedyMappingP(i,t). By assumption and the fact that p_i is good, every point of $Q[t,t^*]$ lies within B. Let $r^{se} = \sigma^{-1}(s^e)$ and $r^{te} = \theta^{-1}(t^e)$.

Suppose GreedyMappingP(i,t) does not do a recursive call. If we output the trivial interval $\{(m,n)\}$, then the lemma is trivially true. Suppose we do not output the trivial interval and $r^{se} \leq r^{te}$. Point $Q(\theta(r^{se}))$ lies on an edge $Q[j_k - 1, j_k]$ with one of the points $Q(t_k)$ where $\mathtt{d}(P(s^e), Q(t_k)) \leq \delta$. By definition of t_k, we have $t_k \leq \theta(r^{se})$. Recall, the free space is convex within each individiaul cell of the free space diagram [6]. Therefore, the set of $s^e \leq s' \leq i^e$ such that $\mathsf{FD}(P[s^e, s'], Q[\theta(r^{se}), j_k]) \leq \delta$ is precisely the approximate reachability interval $[s_k^a, s_k^b] \times \{j_k\}$ we computed. Similarly, the set of $\theta(r^{se}) \leq t' \leq j_k$ such that $\mathsf{FD}(P[s^e, i^e], Q[\theta(r^{se}), t']) \leq \delta$ is actually a *suffix* of the approximate reachability interval $\{i^e\} \times [t_k^a, t_k^b]$ we computed. A similar argument applies if $r^{te} < r^{se}$.

Finally, suppose GreedyMappingP(i,t) recursively calls GreedyMappingP(i^e, t^f). Let t^{f*} be matched with i^e and t^{c*} be matched with $i^e - 1$ by (σ, θ). Because p_{i^e-1} and p_{i^e} are both good, $\mathtt{d}(p_{i^e-1}, Q(t^{c*})) \leq \delta$, and $\mathtt{d}(p_{i^e}, Q(t^f)) \leq \delta$, points $Q(t^{c*})$ and $Q(t^f)$ lie within the same boxes as p_{i^e-1} and p_{i^e}, respectively. These boxes are distinct, so we may conclude $t^{c*} \leq t^f$. Further, we chose $t^c \geq t^{c*}$ and $t^f \leq t^{f*}$, and we may infer $Q(t^c)$ and $Q(t^{f*})$ also lie in the same boxes as p_{i^e-1} and p_{i^e}, respectively. We conclude $t^{c*} \leq t^c < t^f \leq t^{f*}$.

Consider the following correspondence between $P[i^e - 1, i^e]$ and $Q[t^c, t^f]$: Let $s^c \geq i^e - 1$ and $s^f \leq i^e$ be matched to t^c and t^f, respectively, by (σ, θ). We match every point of $P[i^e - 1, s^c]$ to $Q(t^c)$, match $P[s^c, s^f]$ to $Q[t^c, t^f]$ exactly as done by (σ, θ), and match every point of $P[s^f, i^e]$ to $Q(t^f)$. See Figure 4. We have $\mathtt{d}(p_{i^e-1}, Q(t^c)) \leq \delta$ and $\mathtt{d}(P(s^c), Q(t^c)) \leq \delta$, so the entire line segment $P[i^e - 1, s^c]$ lies within distance δ of $Q(t^c)$. Similarly, the line segment $P[s^f, i^e]$ lies within distance δ of $Q(t^f)$. Our correspondence has cost at most δ.

Now, consider any point $Q(t')$ with $t^{c*} \leq t' \leq t^c$ and let s' be matched to t' by (σ, θ). We have $\mathtt{d}(P(s'), Q(t')) \leq \delta$. We argued that line segment $P[i^e - 1, s^c]$ is within distance δ of $Q(t^c)$, implying $\mathtt{d}(P(s'), Q(t^c)) \leq \delta$. Finally, $\mathtt{d}(p_{i^e-1}, Q(t^c)) \leq \delta$. By triangle inequality, $\mathtt{d}(p_{i^e-1}, Q(t')) \leq 3\delta$, implying $Q(t')$ lies in B. As explained above, every point of $Q[t, t^*]$ lies in B. Also, every point of $Q[t^*, t^{c*}]$ lies within distance δ of a point in $P[i, i^e - 1]$ and therefore lies in or within distance δ of B. And, we just showed every point

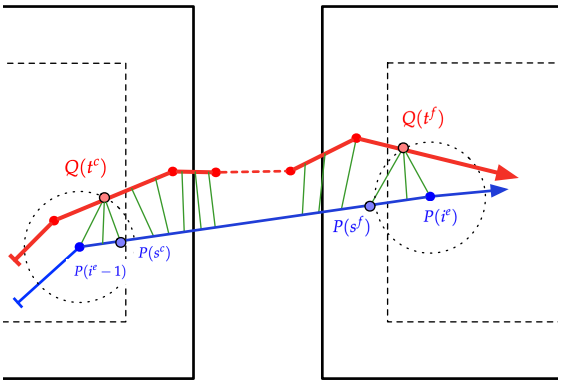


Figure 4 A correspondence of cost δ between $P[i^e-1, i^e]$ and $Q[t^c, t^f]$. A subset of matched points are represented by thin green line segments.

of $Q[t^{c*}, t^c]$ lies in B. Our algorithm will succeed at all its distance checks and recursively call GreedyMappingP(i^e, t^f). Finally, a similar triangle inequality argument implies every point of $Q[t^f, t^{*f}]$ is at most distance 3δ from p_{i^e}. We are inductively guaranteed that (σ, θ) passes through an approximate reachability interval output during the recursive call. Similar arguments apply if GreedyMappingP(i, t) does a recursive call GreedyMappingQ(j^e, s^f).

The proof for GreedyMappingQ(j, s) is the same as that given above, but with the roles of P and Q exchanged. ◀

3.2 Remaining decision procedure details

We now fill in the remaining details of our approximate decision procedure. Recall, we have computed a grid G with boxes of side length $\alpha \cdot \delta$ such that there are $O(n/\alpha)$ bad vertices of P and Q. Also recall, p_1, p_m, q_1, and q_n are designated as bad regardless of their position in G's boxes. As described below, our decision procedure, iteratively in lexicographic order, checks each cell of the free space diagram for which we may have computed an approximate reachablity interval on its left or bottom side. We then extend the known approximately reachable space from each non-empty interval in one of two ways. Depending on whether relevant edges are good or bad, we either perform a call to the appropriate greedy mapping subroutine to seek out new intervals that are approximately reachable but potentially far away in the free space diagram, or we directly compute approximate reachability intervals on the right or top sides of the cell using the constant time method of Alt and Godau [6].

Specifically, we first check if $\mathtt{d}(p_1, q_1) \leq \delta$. If not, our procedure reports failure. Otherwise, let t^b and s^b be the maximum values in $[1, 2]$ such that $\mathtt{d}(p_1, Q(t^b)) \leq \delta$ and $\mathtt{d}(P(s^b), q_1) \leq \delta$, respectively. We designate intervals $\{1\} \times [1, t^b]$ and $[1, s^b] \times \{1\}$ as (approximately) reachable. Now, for each $i \in \langle 2, \ldots, m\rangle$ such that p_{i-1} is dangerous, for each $j \in \langle 2, \ldots, n\rangle$ such that q_{j-1} is dangerous, we do the following.

Suppose we have designated an interval $\{i-1\} \times [t^a, t^b]$ as approximately reachable where $j - 1 \leq t^a \leq t^b \leq j$. Suppose edge $P[i-1, i]$ is good. Then, we run the procedure GreedyMappingP$(i-1, t^a)$. If edge $P[i-1, i]$ is bad, we compute new approximate reachability intervals more directly as follows. First, let $t^{a'}$ be the minimum value in $[t^a, j]$ such that $\mathtt{d}(p_i, Q(t^{a'})) \leq \delta$, and let $t^{b'}$ be the maximum value in $[t^a, j]$ such that $\mathtt{d}(p_i, Q(t^{b'})) \leq \delta$. We designate interval $\{i\} \times [t^{a'}, t^{b'}]$ as approximately reachable (again, we may end up extending a previously computed approximately reachability interval on

$\{i\} \times [j-1, j]$). Similarly, let $s^{a'}$ be the minimum value in $[i-1, i]$ such that $\mathtt{d}(P(s^{a'}), q_j) \leq \delta$, and let $s^{b'}$ be the maximum value in $[i-1, i]$ such that $\mathtt{d}(P(s^{a'}), q_j) \leq \delta$. We designate interval $[s^{a'}, s^{b'}] \times \{j\}$ as approximately reachable. We are done working with interval $\{i-1\} \times [t^a, t^b]$.

Now, suppose we have designated interval $[s^a, s^b] \times \{j-1\}$ as approximately reachable where $i-1 \leq s^a \leq s^b \leq i$. Suppose edge $Q[j-1, j]$ is good. If so, we run the procedure GreedyMappingQ$(j-1, s^a)$. If edge $Q[j-1, j]$ is bad, we compute new approximate reachability intervals more directly as follows. First, let $t^{a'}$ be the minimum value in $[j-1, j]$ such that $\mathtt{d}(p_i, Q(t^{a'})) \leq \delta$, and let $t^{b'}$ be the maximum value in $[j-1, j]$ such that $\mathtt{d}(p_i, Q(t^{b'})) \leq \delta$. We designate interval $\{i\} \times [t^{a'}, t^{b'}]$ as approximately reachable. Similarly, let $s^{a'}$ be the minimum value in $[s^a, i]$ such that $\mathtt{d}(P(s^{a'}), q_j) \leq \delta$, and let $s^{b'}$ be the maximum value in $[s^b, i]$ such that $\mathtt{d}(P(s^{a'}), q_j) \leq \delta$. We designate interval $[s^{a'}, s^{b'}] \times \{j\}$ as approximately reachable. We are done working with interval $[s^a, s^b] \times \{j-1\}$.

Once we have completed the iterations, we do one final step. We check if (m, n) lies on an approximate reachability interval. If so, we report there is a Fréchet correspondence between P and Q of cost $O(\alpha) \cdot \delta$. Otherwise, we report failure.

The following lemmas establish the correctness and running time for our decision procedure. Due to space constraints, proofs are given in the full version of the paper.

▶ **Lemma 3.** *The approximate decision procedure creates approximate reachability intervals only between bad edges of P or Q and dangerous vertices of Q or P, respectively.*

▶ **Lemma 4.** *The approximate decision procedure is correct if it reports $\mathsf{FD}(P, Q) \leq O(\alpha) \cdot \delta$.*

▶ **Lemma 5.** *Suppose there exists a Fréchet correspondence (σ, θ) between P and Q of cost at most δ. The approximate decision procedure will report $\mathsf{FD}(P, Q) \leq O(\alpha) \cdot \delta$.*

▶ **Lemma 6.** *Procedures GreedyMappingP(i, t) and GreedyMappingQ(j, s) can be implemented to run in $O(n)$ time.*

▶ **Lemma 7.** *The approximate decision procedure can be implemented to run in $O(n^3/\alpha^2)$ time.*

Our decision procedure is easily extended to actually output a correspondence of cost $O(\alpha) \cdot \delta$ instead of merely determining if one exists by concatenating the smaller correspondences we discover directly during the iterations or during runs of GreedyMappingP and GreedyMappingQ as we compute approximate reachability intervals. We are now able to state the main result of this section.

▶ **Lemma 8.** *Let P and Q be two polygonal chains in $\mathbb{R}^d$ of at most n vertices each, let $\alpha \in [\sqrt{n}, n]$, and let $\delta \geq 0$ be a parameter. We can compute a Fréchet correspondence between P and Q of cost at most $O(\alpha) \cdot \delta$ or verify that $\mathsf{FD}(P, Q) > \delta$ in $O(n^3/\alpha^2)$ time.*

4 The approximation algorithm

We now describe how to turn our approximate decision procedure into an approximation algorithm whose approximation ratio is arbitrarily close to that of the decision procedure. We emphasize that our techniques use the decision procedure as a black box subroutine, so any improvement to the running time of our approximate decision procedure will imply

the same improvement to our approximation algorithm. In short, we use our approximate decision procedure to binary search over a set of $O(n)$ distances approximating the distances between vertices of P and Q. If the Fréchet distance lies in a large enough gap between a pair of these approximate distances, then we can simplify both polygonal chains so that their edge lengths become large compared to their Fréchet distance. We then run an exact Fréchet distance algorithm of Gudmundsson et al. [21] designed for this case.

Let $P : [1, m] \to \mathbb{R}^d$ and $Q : [1, n] \to \mathbb{R}^d$ be two polygonal chains in d-dimensional Euclidean space, and suppose we have an approximate decision procedure for the Fréchet distance between two polygonal chains with approximation ratio α. We assume α is at most a polynomial function of n (although it may be constant). Let $T(n, \alpha)$ denote the worst-case running time of the procedure on two polygonal chains of at most n vertices each. We assume $T(n, \alpha) = \Omega(n)$. Finally, consider any $0 < \varepsilon \leq 1$. We describe how to compute an $O((1 + \varepsilon)\alpha)$-approximation of $\mathsf{FD}(P, Q)$ in $O(T(n, \alpha) \log(n/\varepsilon))$ time.

We begin by performing a binary search over a set Z of $O(n)$ values close to all of the distances between pairs of vertices in P and Q. Let V denote the set of vertex points in P and Q. Our set Z is such that for any pair of distinct points $o_1, o_2 \in V$, there exist $x, x' \in Z$ such that $x \leq \mathsf{d}(o_1, o_2) \leq x' \leq 2x$. Such a set can be computed in $O(n \log n)$ time [18, Lemma 3.9]. To perform the binary search, we simply search "down" if the approximate decision procedure finds an α-approximate correspondence, and we search "up" if it does not. Let a and b be the largest value of Z for which the procedure fails and the smallest value for which it succeeds, respectively. If a does not exist, then we return the correspondence of cost $\alpha \cdot b$ found for b. We are guaranteed b exists, because the maximum distance between P and Q is achieved at a pair of vertices. From here on, we assume a exists.

We check if the approximate decision procedure finds a correspondence when given parameter $\delta := 12a/\varepsilon$. If so, let Z^a denote the sequence of distances $\left\langle (1+\varepsilon)^0 \cdot a, (1+\varepsilon)^1 \cdot a, \ldots, (1+\varepsilon)^{\lceil 12/\varepsilon \rceil} \cdot a \right\rangle$. We binary search over Z^a and return the cheapest correspondence found.

Suppose no correspondence is found for $12a/\varepsilon$. We check if the approximate decision procedure finds a correspondence when given parameter $\delta := b/(2(1+\varepsilon/2)(1+\sqrt{d})\alpha)$. If *not*, let Z^b denote the sequence of distances $\left\langle b/(1+\varepsilon)^0, b/(1+\varepsilon)^1, \ldots, b/(1+\varepsilon)^{\lceil 2(1+\varepsilon/2)(1+\sqrt{d})\alpha \rceil} \right\rangle$. We binary search over Z^b and return the cheapest correspondence found.

Finally, suppose no correspondence is found for $12a/\varepsilon$ but one is found for $b/(2(1 + \varepsilon/2)(1 + \sqrt{d})\alpha)$. We perform a $3a$-simplification of P and Q, yielding the polygonal chains $\hat{P}$ and $\hat{Q}$ with at most n vertices each. Gudmundsson et al. [21] describe an $O(n \log n)$ time algorithm that computes the Fréchet distance of two polygonal chains *exactly* if all of their edges have length at least $(1 + \sqrt{d})$ times their Fréchet distance. Their algorithm will succeed in finding an optimal Fréchet correspondence between $\hat{P}$ and $\hat{Q}$. This correspondence can be modified to create one for P and Q of cost at most $(1+\varepsilon)\alpha \cdot \mathsf{FD}(P, Q)$ (see Driemel et al. [18, Lemmas 2.3 and 3.5]). We prove the following two lemmas in the full version of the paper.

▶ **Lemma 9.** *The approximation algorithm finds a correspondence between P and Q of cost at most $(1 + \varepsilon)\alpha \cdot \mathsf{FD}(P, Q)$.*

▶ **Lemma 10.** *The approximation algorithm can be implemented to run in $O(T(n, \alpha) \log(n/\varepsilon))$ time.*

We may now state the main result of this section.

▶ **Theorem 11.** *Suppose we have an α-approximate decision procedure for Fréchet distance that runs in time $T(n, \alpha)$ on two polygonal chains in $\mathbb{R}^d$ of at most n vertices each. Let $0 < \varepsilon \leq 1$. Given two such chains P and Q, we can find a Fréchet correspondence between P and Q of cost at most $(1 + \varepsilon)\alpha \cdot \mathsf{FD}(P, Q)$ in $O(T(n, \alpha) \log(n/\varepsilon))$ time.*

Combining Theorem 11 with Lemma 8 while setting $\varepsilon := 1$ gives us our main result.

▶ **Corollary 12.** *Let P and Q be two polygonal chains in $\mathbb{R}^d$ of at most n vertices each, and let $\alpha \in [\sqrt{n}, n]$. We can compute a Fréchet correspondence between P and Q of cost at most $O(\alpha) \cdot \mathsf{FD}(P, Q)$ in $O((n^3/\alpha^2) \log n)$ time.*

5 Conclusion

We described the first strongly subquadratic time approximation algorithm for the continuous Fréchet distance that has a subexponential approximation guarantee. Specifically, it computes an $O(\alpha)$-approximate Fréchet correspondence in $O((n^3/\alpha^2) \log n)$ time for any $\alpha \in [\sqrt{n}, n]$. We admit that our result is not likely the best running time one can achieve and that it serves more as a first major step toward stronger results. In particular, we are at a major disadvantage compared to the $O(n \log n + n^2/\alpha^2)$ time algorithm of Chan and Rahmati [15] for discrete Fréchet distance in that they rely on a constant time method for testing subsequences of points for equality and we know of no analogous procedure for quickly testing (near) equality of subcurves. However, it may not be the case that our own running time analysis is even tight; perhaps a more involved analysis applied to a slight modification of our decision procedure could lead to a better running time. We leave open further improvements such as the one described above.

References

1. Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. Tight hardness results for LCS and other sequence similarity measures. In *Proc. 56th Ann. IEEE Symp. Found. Comp. Sci.*, pages 59–78, 2015.
2. Amir Abboud, Thomas Dueholm Hansen, Virginia Vassilevska Williams, and Ryan Williams. Simulating branching programs with edit distance and friends: Or: a polylog shaved is a lower bound made. In *Proc. 48th Ann. ACM Sympos. Theory of Comput.*, pages 375–388, 2016.
3. Pankaj K. Agarwal, Rinat Ben Avraham, Haim Kaplan, and Micha Sharir. Computing the discrete Fréchet distance in subquadratic time. *SIAM J. Comput.*, 43(2):429–449, 2014.
4. Pankaj K. Agarwal, Kyle Fox, Jiangwei Pan, and Rex Ying. Approximating dynamic time warping and edit distance for a pair of point sequences. In *Proc. 32nd Int. Conf. Comput. Geom.*, pages 6:1–6:16, 2016.
5. Mahmuda Ahmed, Sophia Karagiorgou, Dieter Pfoser, and Carola Wenk. *Map Construction Algorithms*. Springer, 2015.
6. Helmut Alt and Michael Godau. Computing the Fréchet distance between two polygonal curves. *Int. J. Comput. Geom. Appl.*, 5:75–91, 1995.
7. Alexandr Andoni and Negev Shekel Nosatzki. Edit distance in near-linear time: it's a constant factor. In *Proc. 61st Ann. IEEE Symp. Found. Comput. Sci.*, 2020.
8. Boris Aronov, Sariel Har-Peled, Christian Knauer, Yusu Wang, and Carola Wenk. Fréchet distance for curves, revisited. In *Proc. 14th Ann. Euro. Sympos. Algo.*, pages 52–63, 2006.
9. Arturs Backurs and Piotr Indyk. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). *SIAM J. Comput.*, 47(3):1087–1097, 2018.
10. Karl Bringmann. Why walking the dog takes time: Fréchet distance has no strongly subquadratic algorithms unless SETH fails. In *Proc. 55th. Ann. IEEE Symp. Found. Comp. Sci.*, pages 661–670, 2014.
11. Karl Bringmann and Marvin Künnemann. Quadratic conditional lower bounds for string problems and dynamic time warping. In *Proc. 56th Ann. IEEE Symp. Found. Comp. Sci.*, pages 79–97, 2015.

12 Karl Bringmann and Marvin Künnemann. Improved approximation for Fréchet distance on c-packed curves matching conditional lower bounds. *Int. J. Comput. Geom. Appl.*, 27(1-2):85–120, 2017.

13 Karl Bringmann and Wolfgang Mulzer. Approximability of the discrete Fréchet distance. *J. Comput. Geom.*, 7(2):46–76, 2016.

14 Kevin Buchin, Maike Buchin, Wouter Meulemans, and Wolfgang Mulzer. Four Soviets walk the dog: Improved bounds for computing the Fréchet distance. *Discrete Comput. Geom.*, 58(1):180–216, 2017.

15 Timothy M. Chan and Zahed Rahmati. An improved approximation algorithm for the discrete Fréchet distance. *Inf. Process. Lett.*, 138:72–74, 2018.

16 Daniel Chen, Anne Driemel, Leonidas J. Guibas, Andy Nguyen, and Carola Wenk. Approximate map matching with respect to the Fréchet distance. In *Proc. 13th Meeting on Algorithm Engineering and Experiments*, pages 75–83, 2011.

17 Richard Cole. Slowing down sorting networks to obtain faster sorting algorithms. *J. ACM*, 34(1):200–208, 1987.

18 Anne Driemel, Sariel Har-Peled, and Carola Wenk. Approximating the Fréchet distance for realistic curves in near linear time. *Discrete Comput. Geom.*, 48(1):94–127, 2012.

19 Kyle Fox and Xinyi Li. Approximating the geometric edit distance. In *Proc. 30th Int. Symp. Algo. Comput.*, pages 26:1–26:16, 2019.

20 Omer Gold and Micha Sharir. Dynamic time warping and geometric edit distance: Breaking the quadratic barrier. *ACM Trans. Algorithms*, 14(4):50:1–50:17, 2018.

21 Joachim Gudmundsson, Majid Mirzanezhad, Ali Mohades, and Carola Wenk. Fast Fréchet distance between curves with long edges. *Int. J. Comput. Geom. Appl.*, 29(2):161–187, 2019.

22 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *J. Comp. Sys. Sci.*, 62(2):367–375, 2001.

23 Minghui Jiang, Ying Xu, and Binhai Zhu. Protein structure-structure alignment with discrete Fréchet distance. *J. Bioinformatics and Computational Biology*, 6(1):51–64, 2008.

24 William Kuszmaul. Dynamic time warping in strongly subquadratic time: Algorithms for the low-distance regime and approximate evaluation. In *Proc. 46th Intern. Colloqu. Automata, Languages, Programming*, pages 80:1–80:15, 2019.

25 Nimrod Megiddo. Applying parallel computation algorithms in the design of serial algorithms. *J. Assoc. Comput. Mach.*, 30(4):852–865, 1983.

26 Majid Mirzanezhad. On the approximate nearest neighbor queries among curves under the Fréchet distance. *CoRR*, abs/2004.08444, 2020. `arXiv:2004.08444`.

27 E. Sriraghavendra, K. Karthik, and Chiranjib Bhattacharyya. Fréchet distance based approach for searching online handwritten documents. In *Proc. 9th Intern. Conf. Document Analysis and Recognition*, pages 461–465, 2007.

28 Rex Ying, Jiangwei Pan, Kyle Fox, and Pankaj K. Agarwal. A simple efficient approximation algorithm for dynamic time warping. In *Proc. 24th ACM SIGSPATIAL Int. Conf. Adv. Geo. Inf. Sys.*, 2016.

Computing the Multicover Bifiltration

René Corbet
Department of Mathematics, KTH Royal Institute of Technology, Stockholm, Sweden

Michael Kerber
Institute of Geometry, Graz University of Technology, Austria

Michael Lesnick
Department of Mathematics and Statistics, University at Albany, SUNY, NY, USA

Georg Osang
IST Austria (Institute of Science and Technology Austria), Klosterneuburg, Austria

Abstract

Given a finite set $A \subset \mathbb{R}^d$, let $\mathrm{Cov}_{r,k}$ denote the set of all points within distance r to at least k points of A. Allowing r and k to vary, we obtain a 2-parameter family of spaces that grow larger when r increases or k decreases, called the *multicover bifiltration*. Motivated by the problem of computing the homology of this bifiltration, we introduce two closely related combinatorial bifiltrations, one polyhedral and the other simplicial, which are both topologically equivalent to the multicover bifiltration and far smaller than a Čech-based model considered in prior work of Sheehy. Our polyhedral construction is a bifiltration of the *rhomboid tiling* of Edelsbrunner and Osang, and can be efficiently computed using a variant of an algorithm given by these authors as well. Using an implementation for dimension 2 and 3, we provide experimental results. Our simplicial construction is useful for understanding the polyhedral construction and proving its correctness.

2012 ACM Subject Classification Mathematics of computing → Combinatorial algorithms; Theory of computation → Computational geometry

Keywords and phrases Bifiltrations, nerves, higher-order Delaunay complexes, higher-order Voronoi diagrams, rhomboid tiling, multiparameter persistent homology, denoising

Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.27

Related Version *Full Version*: `https://arxiv.org/abs/2103.07823`

Funding The first two authors were supported by the Austrian Science Fund (FWF) grant number P 29984-N35 and W1230. The first author was partly supported by an Austrian Marshall Plan Scholarship, and by the Brummer & Partners MathDataLab.

Acknowledgements The authors want to thank the reviewers for many helpful comments and suggestions.

1 Introduction

Let A be a finite subset of $\mathbb{R}^d$, whose points we call *sites*. For $r \in [0, \infty)$ and an integer $k \geq 0$, we define

$$\mathrm{Cov}_{r,k} := \left\{ b \in \mathbb{R}^d \mid \; ||b - a|| \leq r \text{ for at least } k \text{ sites } a \in A \right\}.$$

Thus, $\mathrm{Cov}_{r,k}$ is the union of all k-wise intersections of closed balls of radius r centered at the sites; see Figure 1. Define a *bifiltration* to be a collection of sets

$$C := (C_{r,k})_{(r,k)\in[0,\infty)\times\mathbb{N}}$$

such that $C_{r,k} \subseteq C_{r',k'}$ whenever $r \leq r'$ and $k \geq k'$. Clearly, the sets

$$\mathrm{Cov} := (\mathrm{Cov}_{r,k})_{(r,k)\in[0,\infty)\times\mathbb{N}}$$

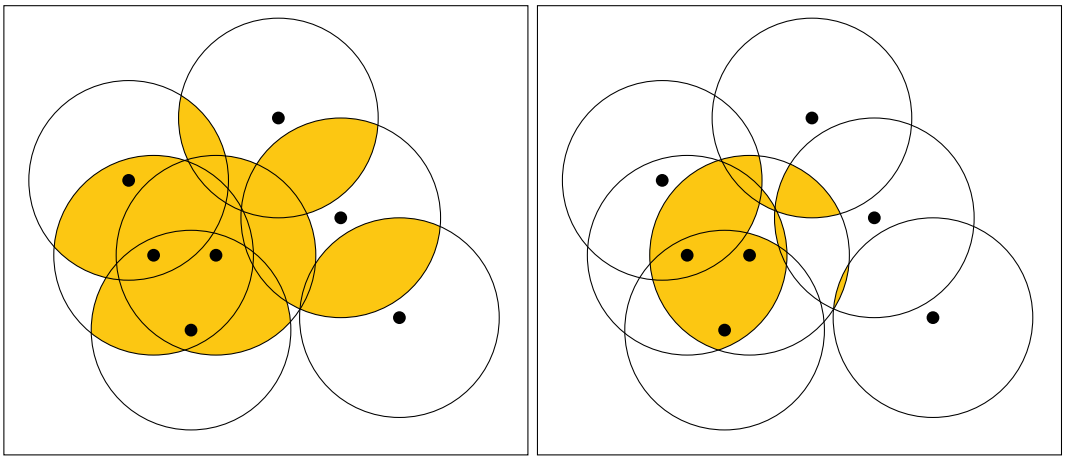


Figure 1 The 2- and 3-fold cover of a few points with respect to a certain radius. The first homology of the 2-fold cover is trivial, while the first homology of the 3-fold cover is non-trivial.

form a bifiltration. This is known as the *multicover bifiltration*. It arises naturally in topological data analysis (TDA), and specifically, in the topological analysis of data with outliers or non-uniform density [17, 26, 44].

We wish to study the topological structure of the bifiltration Cov algorithmically in practical applications, via 2-parameter persistent homology [10]. For this, the natural first step is to compute a *combinatorial model* of Cov, that is, a purely combinatorial bifiltration C which is topologically equivalent to Cov. This step is the focus of the present paper. For computational efficiency, C should not be too large.

In fact, we propose two closely related combinatorial models C, one polyhedral and one simplicial. The polyhedral model is a bifiltration of the *rhomboid tiling*, a polyhedral cell complex in $\mathbb{R}^{d+1}$ recently introduced by Edelsbrunner and Osang to study the multicover bifiltration [26]. Edelsbrunner and Osang have given an efficient algorithm for computing the rhomboid tiling [27], and this adapts readily to compute our bifiltration. We use the simplicial model to prove that the polyhedral model is topologically equivalent to Cov.

Motivation and prior work. For $k = 1$ fixed, $(\mathrm{Cov}_{r,1})_{r\in[0,\infty)}$ is the well-known *offset filtration* (also known as the union of balls filtration), a standard construction for analyzing the topology of a finite point sample across scales [25]. It is a central object in the field of persistent homology. While the persistent homology of this filtration is stable to small geometric perturbations of the sites [19], it is not robust with respect to outliers, and it can be insensitive to topological structure in high density regions of the data.

Within the framework of 1-parameter persistent homology, there have been many proposals for alternative constructions which address these issues. These approaches include the removal of low density outliers [9], filtering by a density function [7, 14, 15], distance to measure constructions [1, 8, 17, 29], kernel density functions [42], and subsampling [4]. A detailed overview of these approaches can be found in [6].

Several of these constructions have good stability properties or good asymptotic behavior. However, as explained in [6], all of the known 1-parameter persistence strategies for handling outliers or variations in density share certain disadvantages: First, they all depend on a choice of a parameter. Typically, this parameter specifies a fixed spatial scale or a density threshold at which the construction is carried out. In the absence of a priori knowledge

about the structure of the data, it may be unclear how to select such a parameter. And if the data exhibits topological features at multiple spatial or density scales, it may be that no single parameter choice allows us to capture all the structure present in the data. Second, constructions that fix a scale parameter are unable to distinguish between small spatial features and large ones, and constructions that fix a density or measure parameter are unable to distinguish features in densely sampled regions of the data from features involving sparse regions.

A natural way to circumvent these limitations is to consider a 2-parameter approach, where one constructs a bifiltration from the data, rather than a 1-parameter filtration [10]. The multicover bifiltration is one natural option for this. Alternatives include the density bifiltrations of Carlsson and Zomorodian [10], and the degree bifiltrations of Lesnick and Wright [39]; again, we refer the reader to [6] for a more detailed discussion. Among these three options, the multicover bifiltration has two attractive features which together distinguish it from the others. First, its construction does not depend on any additional parameters. Second, the multicover bifiltration satisfies a strong stability property, which in particular guarantees robustness to outliers [6].

There is a substantial and growing literature on the use of bifiltrations in data analysis. Most approaches begin by applying homology with coefficients to the bifiltration, to obtain an algebraic object called *bipersistence module*. In contrast to the 1-parameter case, where the algebraic structure of a persistence module is completely described by a barcode [49], it is well-known that defining the barcode of bipersistence modules is problematic [10]. Nevertheless, one can compute invariants of a bipersistence module which serve as useful surrogates for a barcode, and a number of ideas for this have been proposed [10, 13, 30, 39, 47].

Regardless of which invariants of the multicover bifiltration we wish to consider, to work with this bifiltration computationally, the natural first step is to find a reasonably sized combinatorial (i.e., simplicial or polyhedral) model for the bifiltration. With such a model, recently developed algorithms such as those described in [40] and [34] can efficiently compute minimal presentations and standard invariants of the homology modules of the bifiltration.

In the 1-parameter setting, there are two well-known simplicial models of the offset filtration. The *Čech filtration*, is given at each scale by the nerve of the balls; the equivalence of the offset and Čech filtrations follows from the *Persistent Nerve Theorem*. For large point sets, the full Čech filtration is too large to be used in practical computations. However, the *alpha filtration* (also known as the *Delaunay filtration*) [23, 25] is a much smaller subfiltration of the Čech filtration which is also simplicial model for the offset filtration. It is given at each scale by intersecting each ball with the Voronoi cell [48] of its center, and then taking the nerve of the resulting regions. For d small (say $d \leq 3$), the Delaunay filtration is readily computed in practice for many thousands of points.

It is implicit in the work of Sheehy [44] that the multicover bifiltration has an elegant simplicial model, the *subdivision-Čech bifiltration*, obtained via a natural filtration on the barycentric subdivision of each Čech complex; see also [11, Appendix B] and [6, Section 4]. However, the subdivision-Čech bifiltration has exponentially many vertices in the size of the data, making it even more unsuitable for computations than the ordinary Čech filtration.

Edelsbrunner and Osang [26] therefore seek to develop the computational theory of the multicover bifiltration using *higher-order Delaunay complexes* [24, 36], taking the alpha filtration as inspiration. Assuming the sites are in general position, they define a polyhedral cell complex in $\mathbb{R}^{d+1}$ called the *rhomboid tiling*, which contains all higher-order Delaunay complexes as planar sections. Using the rhomboid tiling, they present a polynomial time algorithm to compute the barcodes of a horizontal or vertical slice of the multicover bifiltration (i.e., of a one-parameter filtration obtained by fixing either one of the two parameters r, k).

The case of fixed r and varying k is more challenging because the order-k Delaunay complexes do not form a filtration. The authors construct a zigzag filtration for this case. The problem of efficiently computing 2-parameter persistent homology of the multicover bifiltration is not addressed by [26].

We note that the subdivision-Čech bifiltration is more general than the Rhomboid tiling: the rhomboid tiling is defined only for Euclidean data, whereas the topological equivalence of the subdivision-Čech and multicover bifiltrations extends to data in any metric space where finite intersections of balls are contractible.

Contributions. We introduce the first efficiently computable combinatorial models of the multicover bifiltration Cov. First, we introduce a simplicial model, whose construction is based on two main ideas: In order to connect the higher-order alpha complex constructions for (r, k) and $(r, k+1)$, we simply overlay their underlying covers to a "double-cover", whose nerve is a simplicial complex that contains both alpha complexes. This yields a zigzag of simplicial filtrations. The second main idea is that this zigzag can be "straightened out" to a (non-zigzaging) bifiltration, simply by taking unions of prefixes in the zigzag sequence. This straightening technique has previously been used by Sheehy to construct sparse approximations of Vietoris-Rips complexes [45]. Together, these two ideas give rise to a bifiltration S-Del $:= (\text{S-Del}_{r,k})_{(r,k)\in[0,\infty)\times\mathbb{N}}$ of simplicial complexes.

The bifiltration S-Del can also be obtained directly as the persistent nerve of a "thickening" of Cov constructed via mapping telescopes. This observation leads to a simple proof of topological equivalence (i.e., weak equivalence; see Section 2) of Cov and S-Del via the Nerve Theorem. It follows that the persistent homology modules of Cov and S-Del are isomorphic.

Our second contribution is to show that the rhomboid tiling as defined in [26] also gives rise to a (non-zigzaging) bifiltration of polyhedral complexes that is topologically equivalent to the multicover. We proceed in two steps: First, we slice every rhomboid at each integer value k (slightly increasing the number of cells) and adapt the straightening trick used to construct S-Del. We prove the topological equivalence of this construction with the multicover bifiltration by relating the slice rhomboid filtration with S-Del. The main observation is a one-to-one correspondence of maximal-dimensional cells in both constructions, which leads to a proof via the Nerve Theorem. Second, we relate the sliced and unsliced rhomboid tilings at every scale via a deformation retraction.

We give size bounds for both of the bifiltrations we introduce. For n points in $\mathbb{R}^d$, we show that their size is $O(n^{d+1})$. This is a decisive improvement over Sheehy's Čech-based construction, which has exponential dependence on n.

An efficient algorithm for computing rhomboid tilings has recently been presented in [27]; hence, using the accompanying implementation RHOMBOIDTILING of this algorithm, our second contribution gives us an efficient software to compute a bifiltration of cell complexes equivalent to Cov, currently for points in $\mathbb{R}^2$ and $\mathbb{R}^3$. We combine this implementation with the libraries MPFREE and RIVET to demonstrate that minimal presentations of multicover persistent homology modules can now be efficiently computed, often within seconds, as can invariants such as the Hilbert function.

2 Background

Filtrations. For P a poset, define a (P-indexed) filtration to be a collection of topological spaces $X = (X_p)_{p\in P}$ indexed by P, such that $X_p \subseteq X_q$ whenever $p \leq q \in P$. For example, an $\mathbb{N}$-indexed filtration X is a diagram of spaces and inclusions of the following form:

$$X_1 \hookrightarrow X_2 \hookrightarrow X_3 \hookrightarrow X_4 \hookrightarrow \cdots .$$

A *morphism* $\varphi : X \to Y$ of P-indexed filtrations is a collection of continuous maps $(\varphi_p : X_p \to Y_p)_{p\in P}$ which commute with the inclusions in X and Y. In the language of category theory, a P-indexed filtration is a functor $P \to \mathbf{Top}$ whose internal maps are inclusions, and a morphism is a natural transformation.

Recall that the product poset $P \times Q$ of posets P and Q is defined by taking $(p, q) \leq (p', q')$ if and only if $p \leq p'$ and $q \leq q'$. When P is the product of two totally ordered sets, we call a P-indexed filtration a *bifiltration*.

In the classical homotopy theory of diagrams of spaces, there is a standard analogue of the notion of homotopy equivalence for diagrams of spaces, called *weak equivalence*. We now define a version of this for P-indexed filtrations: A morphism of filtrations $\varphi : X \to Y$ is called an *objectwise homotopy equivalence* if each $\varphi_p : X_p \to Y_p$ is a homotopy equivalence. If there exists a finite zigzag diagram of objectwise homotopy equivalences

$$X \to Z_1 \leftarrow Z_2 \to \cdots \leftarrow Z_{n-1} \to Z_n \leftarrow Y$$

connecting X and Y, then we say that X and Y are *weakly equivalent*. The terminology originates from the theory of model categories [21, 32]. See [5, 37, 43] for discussions of weak equivalence of diagrams in the context of TDA.

▶ Remark 1. To motivate the consideration of zigzags in the definition above, we note that for X and Y a pair of weakly equivalent P-indexed filtrations, there is not necessarily an objectwise homotopy equivalence $f : X \to Y$. For example, let $P = \mathbb{R}$, X be the offset filtration on $\{0, 1\} \subset \mathbb{R}$, and Y be the nerve filtration of X. It is easy to check that there is no objectwise homotopy equivalence $f : X \to Y$. On the other hand, it follows from the Persistent Nerve Theorem (Theorem 3 below) that X and Y are weakly equivalent. Moreover, one can construct a similar example of weakly equivalent filtrations for which there is no objectwise homotopy equivalence in either direction.

An objectwise weak equivalence $\varphi : X \to Y$ induces isomorphisms on the persistent homology modules of X and Y. Hence, weakly equivalent filtrations have isomorphic persistent homology modules.

We say a P-indexed filtration X is *Euclidean* if $X_p \subset \mathbb{R}^n$ for some n and all $p \in P$.

The Persistent Nerve Theorem. A *cover* of $X \subset \mathbb{R}^n$ is a collection $\mathfrak{X} = \{\mathfrak{X}^i\}_{i\in I}$ of subsets of X whose union is X. The *nerve* of $\mathfrak{X}$ is the abstract simplicial complex

$$\mathrm{Nrv}\,(\mathfrak{X}) := \left\{ \sigma \subset I \mid \bigcap_{i\in\sigma} \mathfrak{X}^i \neq \emptyset \right\}.$$

We say that the cover is *good* if it is finite and consists of closed, convex sets [25].

One version of the *Nerve Theorem* asserts that X and $\mathrm{Nrv}\,(\mathfrak{X})$ are homotopy equivalent whenever $\mathfrak{X}$ is a good cover of X [25, 38]. It is TDA folklore that this version of the Nerve Theorem can be extended to a persistent version; a proof appears in [3]; see also [16] for formulation of the Persistent Nerve Theorem in terms of open covers.

In order to state the Persistent Nerve Theorem for closed, convex covers, we first extend the definition of a cover.

▶ **Definition 2** (Cover of a filtration). *Let P be a poset and X a P-indexed Euclidean filtration. A* cover of X *is a collection $\mathfrak{X} = \{\mathfrak{X}^i\}_{i\in I}$ of P-indexed filtrations such that for each $p \in P$, $\{\mathfrak{X}^i_p \mid i \in I\}$ is a cover of $\mathfrak{X}_p$. We say $\mathfrak{X}$ is good if each $\mathfrak{X}_p := \{\mathfrak{X}^i_p \mid i \in I\}$ is a good cover.*

The definition of the nerve above extends immediately to yield a *nerve filtration* $\mathrm{Nrv}\,(\mathfrak{X})$ associated to any cover of a filtration.

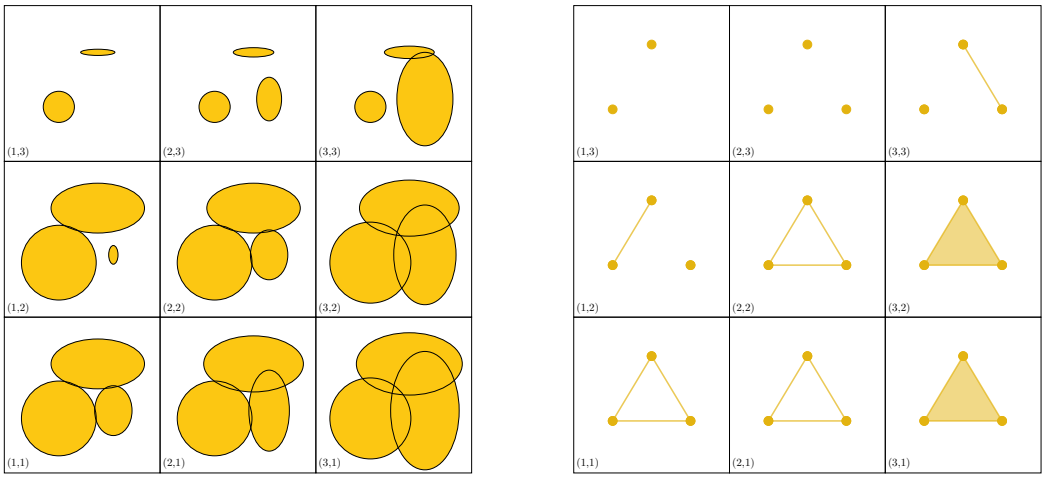


Figure 2 *Left:* A bifiltration of good covers over $\{1 < 2 < 3\} \times \{3 < 2 < 1\}$. *Right:* A bifiltration consisting of the nerves of the covers. The Persistent Nerve Theorem ensures that not only the individual spaces at scales (n, m) are homotopy equivalent, but also that the two bifiltrations are weakly equivalent.

▶ **Theorem 3** (Persistent Nerve Theorem [3]). *Let P be a poset, X a P-indexed Euclidean filtration, and $\mathfrak{X}$ a good cover of X. There exists a diagram of objectwise homotopy equivalences*

$$X \xleftarrow{\simeq} \Delta\mathfrak{X} \xrightarrow{\simeq} \mathrm{Nrv}(\mathfrak{X}).$$

As shown in [3], the intermediate filtration $\Delta\mathfrak{X}$ in the statement of the theorem can be taken to be a homotopy colimit of a diagram constructed from $\mathfrak{X}$, just as in the proof of the Persistent Nerve Theorem for open covers [16]. Note that if P is a singleton set, the Persistent Nerve Theorem specializes to the classical version of the Nerve Theorem mentioned above.

Multicovers. As indicated in the introduction, the multicover bifiltration of a finite set $A \subset \mathbb{R}^d$ is the $\mathbb{R} \times \mathbb{N}^{\mathrm{op}}$-indexed bifiltration Cov given by

$$\mathrm{Cov}_{r,k} := \left\{ b \in \mathbb{R}^d \mid \ ||b - a|| \leq r \text{ for at least } k \text{ points } a \in A \right\}.$$

A first step towards constructing a simplicial model for Cov is to identify a good cover for $\mathrm{Cov}_{r,k}$, with r and k fixed. For $\widetilde{A} \subset A$ we define

$$\mathrm{Cov}_r(\widetilde{A}) := \left\{ b \in \mathbb{R}^d \mid \ ||b - \tilde{a}|| \leq r \text{ for all } \tilde{a} \in \widetilde{A} \right\}.$$

Clearly, $\mathrm{Cov}_{r,k}$ is the union of all $\mathrm{Cov}_r(\widetilde{A})$ such that $\widetilde{A} \subset A$ and $|\widetilde{A}| = k$, and each $\mathrm{Cov}_r(\widetilde{A})$ is closed and convex. Hence, denoting

$$\mathfrak{Cov}_{r,k} := \left\{ \mathrm{Cov}_r(\widetilde{A}) \mid \widetilde{A} \subset A, |\widetilde{A}| = k \right\}$$

and applying the Nerve Theorem, $\mathrm{Cov}_{r,k}$ is homotopy equivalent to $\mathrm{Nrv}(\mathfrak{Cov}_{r,k})$. For fixed k and $r \leq r'$, we have an inclusion $\mathrm{Nrv}(\mathfrak{Cov}_{r,k}) \hookrightarrow \mathrm{Nrv}(\mathfrak{Cov}_{r',k})$.

Note that these nerves can be quite large: for large enough r, $\mathrm{Nrv}(\mathfrak{Cov}_{r,k})$ contains $\binom{|A|}{k}$ vertices. To obtain a smaller simplicial model of $\mathrm{Cov}_{r,k}$, we use the generalization of Delaunay triangulations to higher-order Delaunay complexes. For a subset $\tilde{A} \subset A$ with $|\tilde{A}| = k$, define its *order-k Voronoi region* as

$$\mathrm{Vor}(\tilde{A}) := \left\{ b \in \mathbb{R}^d \mid \ ||b - \tilde{a}|| \leq ||b - a|| \text{ for all } \tilde{a} \in \tilde{A}, a \in A \setminus \tilde{A} \right\}.$$

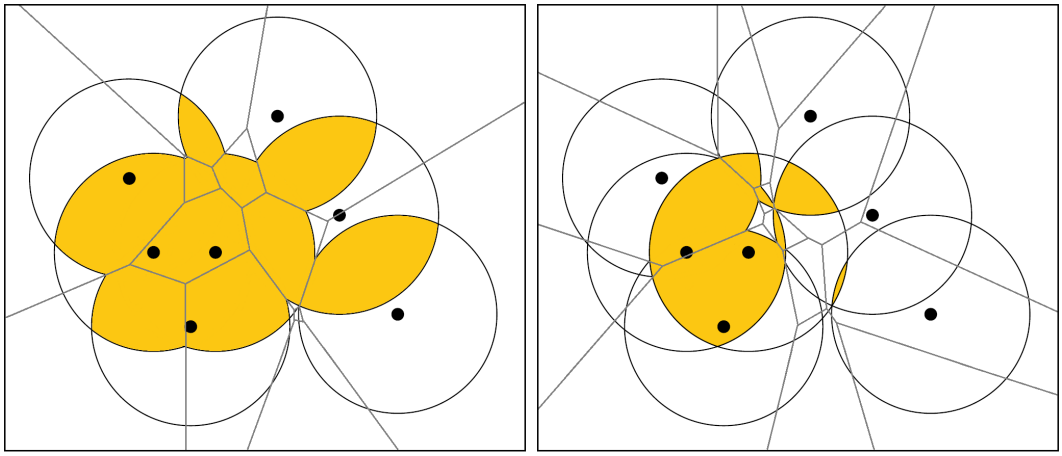


Figure 3 *Left:* The 2-fold cover of some points with respect to a certain radius overlapped with its Voronoi diagram of order 2. $\mathfrak{Vor}$ is combinatorially simpler than $\mathfrak{Cov}$. *Right:* The corresponding 3-fold cover overlapped with its Voronoi diagram of order 3.

The set of all order-k Voronoi regions yield a decomposition of $\mathbb{R}^d$ into closed convex subsets having the same k closest points of A in common. This decomposition is called the *order-k Voronoi diagram* [2, 28]. We denote it as Vor_k.

For any $r \in \mathbb{R}$ and $k \in \mathbb{N}$, intersecting each order-k Voronoi region with the corresponding multicovered region of fixed radius r yields the following good cover of $\mathrm{Cov}_{r,k}$.

$$\mathfrak{Vor}_{r,k} := \left\{ \mathrm{Cov}_r(\tilde{A}) \cap \mathrm{Vor}(\tilde{A}) \mid \tilde{A} \subset A, |\tilde{A}| = k \right\}.$$

For an illustration, see Figure 3. The nerve of $\mathfrak{Vor}_{r,k}$, which we will denote $\mathrm{Del}_{r,k}$, is called an *order-k Delaunay complex*. By the Nerve Theorem, $\mathrm{Del}_{r,k}$ and $\mathrm{Cov}_{r,k}$ are homotopy equivalent. Note that $\mathrm{Del}_{r,1}$ is the *alpha complex* of radius r [23, 25]. A different but related concept is the *order-k Delaunay mosaic*, which is the geometric dual of the order-k Voronoi diagram [26]; see Section 4.

3 A simplicial Delaunay bifiltration

For fixed $r \geq 0$, we have inclusions

$$\cdots \hookrightarrow \mathrm{Cov}_{r,5} \hookrightarrow \mathrm{Cov}_{r,4} \hookrightarrow \mathrm{Cov}_{r,3} \hookrightarrow \mathrm{Cov}_{r,2} \hookrightarrow \mathrm{Cov}_{r,1},$$

but there are no analogous inclusions $\mathrm{Del}_{r,k} \hookrightarrow \mathrm{Del}_{r,k-1}$ between the higher-order Delaunay complexes. Indeed, we do not even have inclusions of the vertex sets. Consequently, $\mathfrak{Vor}$ is not a bifiltration of covers.

Overlaying consecutive covers. We replace the higher-order Delaunay complexes $\mathrm{Del}_{r,k}$ by the nerve of the union of two consecutive $\mathfrak{Vor}_{r,k}$. For an illustration of the outcome, see Figure 4. Formally, we define

$$\widetilde{\mathrm{Del}}_{r,k} := \mathrm{Nrv}\left(\mathfrak{Vor}_{r,k} \cup \mathfrak{Vor}_{r,k+1}\right).$$

Importantly, $\mathfrak{Vor}_{r,k} \cup \mathfrak{Vor}_{r,k+1}$ covers the same space as $\mathfrak{Vor}_{r,k}$ for any $r \geq 0$ and $k \in \mathbb{N}$. Furthermore, we get inclusions

$$\mathrm{Del}_{r,k+1} \hookrightarrow \widetilde{\mathrm{Del}}_{r,k} \hookleftarrow \mathrm{Del}_{r,k}.$$

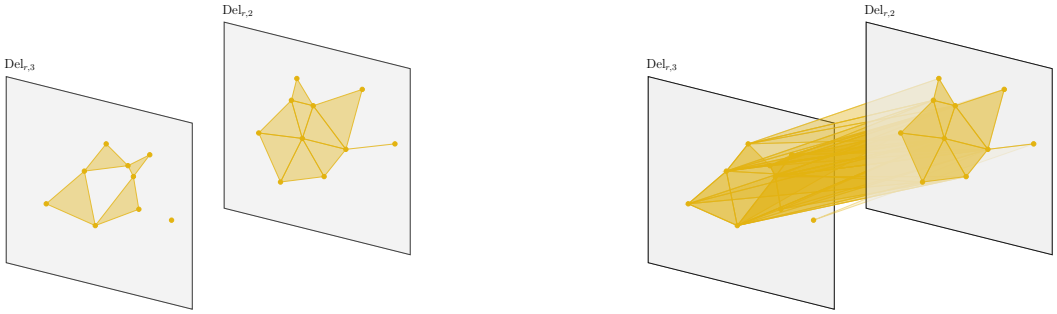


Figure 4 *Left:* The Delaunay complexes of order 2 and 3 of our running example. *Right:* The construction of the simplicial complex $\widetilde{\mathrm{Del}}_{r,2}$. $\widetilde{\mathrm{Del}}_{r,2}$ consists of the Delaunay complexes $\mathrm{Del}_{r,2}$ and $\mathrm{Del}_{r,3}$, and additional mixed simplices connecting these. This connection arises from intersections of the 2-, and 3-fold covers restricted to their Voronoi diagrams of order 2 and 3, respectively.

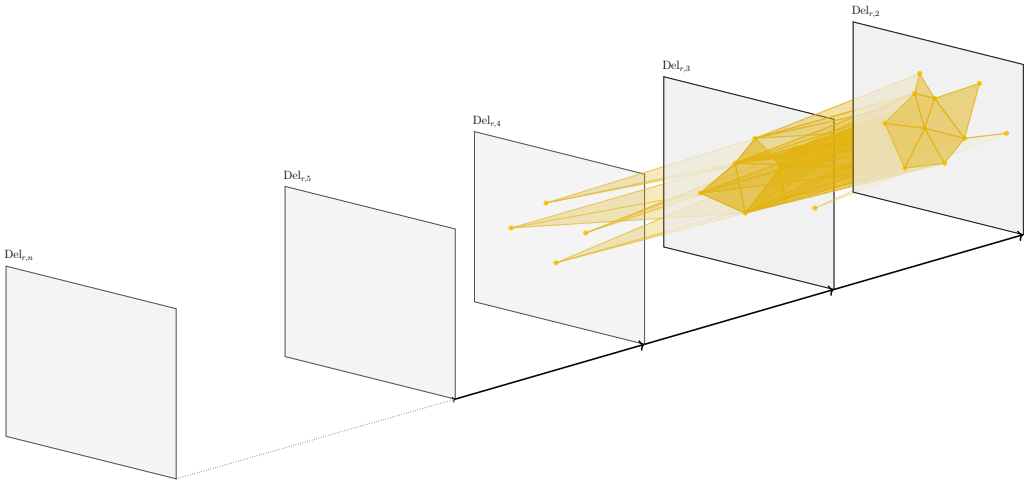


Figure 5 S-Del$_{r,2}$ is union of all $\widetilde{\mathrm{Del}}_{r,i}$, $i \geq 2$. By Theorem 4, it is homotopy equivalent to $\mathrm{Del}_{r,2}$. S-Del is bounded by n, the number of sites in A. Thus, in this case, $\mathrm{Del}_{r,i}$ is empty for $i \geq 5$.

Straightening out zigzags. In order to define a simplicial bifiltration, let $n := |A|$, $r \geq 0$, and define the filtration

$$\mathrm{Del}_{r,n} \hookrightarrow \widetilde{\mathrm{Del}}_{r,n-1} \hookrightarrow \widetilde{\mathrm{Del}}_{r,n-2} \cup \widetilde{\mathrm{Del}}_{r,n-1} \hookrightarrow \cdots \hookrightarrow \bigcup_{i=1}^{n-1} \widetilde{\mathrm{Del}}_{r,i}.$$

Since $\mathrm{Del}_{r,k} = \emptyset$ for all $k > n$, we can denote the spaces in this filtration by

$$\text{S-Del}_{r,k} := \bigcup_{i \geq k} \widetilde{\mathrm{Del}}_{r,i}.$$

For an illustration, see Figure 5. Letting both r and k vary, we obtain a $\mathbb{R} \times \mathbb{N}^{\mathrm{op}}$-indexed bifiltration S-Del. Note that S-Del$_{r,k}$ is *not* equal to the nerve of the union of all $\mathfrak{Vor}_{r,i}$, $i \geq k$, which is a much larger object.

The bifiltration S-Del is our desired simplicial model of the multicover bifiltration:

▶ **Theorem 4.** *The multicover bifiltration* Cov *is weakly equivalent to* S-Del.

The proof of Theorem 4 will use the following variant of the usual mapping telescope construction [31, Section 3.F]: given any sequence of continuous maps

$$C = (C_1 \hookrightarrow C_2 \hookrightarrow \cdots \hookrightarrow C_n),$$

let M_C denote the quotient of the disjoint union

$$\sqcup_{i=1}^{n} C_i \times I$$

given by gluing each $C_i \times \{1\}$ to $C_{i+1} \times \{0\}$ along the inclusion $C_i \hookrightarrow C_{i+1}$. It is easy to check that we have a deformation retraction $M_C \to C_n$.

Proof of Theorem 4. Our proof strategy is similar to ones used for sparse filtrations [12, 46]. We will observe that S-Del is isomorphic to the nerve of a good cover of a bifiltration X which is weakly equivalent to Cov. The result then follows from the Persistent Nerve Theorem.

Let $X_{r,k}$ be the mapping telescope of the sequence of

$$\mathrm{Cov}_{r,n} \hookrightarrow \mathrm{Cov}_{r,n-1} \hookrightarrow \cdots \hookrightarrow \mathrm{Cov}_{r,k}.$$

Letting r and k vary, the spaces $X_{r,k}$ assemble into an $\mathbb{R} \times \mathbb{N}^{\mathrm{op}}$-indexed bifiltration X, and the deformation retractions $X_{r,k} \to \mathrm{Cov}_{r,k}$ assemble into an objectwise homotopy equivalence $X \to \mathrm{Cov}$. Noting that the cover $\mathfrak{Vor}_{r,k}$ of each $\mathrm{Cov}_{r,k}$ induces a cover $\mathfrak{Vor}_{r,k} \times I$ of $\mathrm{Cov}_{r,k} \times I$ in the obvious way, we see that the covers $\{\mathfrak{Vor}_{r,i} \times I\}_{i=k}^{n}$ descend to a good cover of $X_{r,k}$, and these in turn assemble into a good cover of X. It is easy to see that S-Del is isomorphic to the persistent nerve of this cover. ◀

Truncations of S-Del. When the point cloud is large, it may be computationally difficult to construct the full bifiltration S-Del. We instead consider, for $K \in \mathbb{N}$, the bifiltration $\text{S-Del}^{\leq K}$ constructed in the same way, but disregarding all order-k Voronoi cells with $k > K$,

$$\text{S-Del}^{\leq K}_{r,k} := \bigcup_{(K-1) \geq i \geq k} \widetilde{\mathrm{Del}}_{r,i}.$$

Note that $\text{S-Del}^{\leq |A|} = \text{S-Del}$. Viewing $\text{S-Del}^{\leq K}$ as an $(\mathbb{R} \times \{1, \ldots, K\})$-indexed bifiltration, the proof of Theorem 4 adapts immediately to show that $\text{S-Del}^{\leq K}$ is weakly equivalent to the restriction of Cov to $\mathbb{R} \times \{1, \ldots, K\}$.

Size of S-Del. By the *size* of a bifiltration X, we mean the number of simplices in the largest simplicial complex in X. If the sites $A \subset \mathbb{R}^d$ are not in general position, the size of $\text{S-Del}^{\leq K}$ can be huge; indeed, if all points of A lie on a circle in $\mathbb{R}^2$ and r is at least the radius of this circle, then $\mathrm{Del}_{r,k}$ has $2^{\binom{|A|}{k}}$ simplices, so $\text{S-Del}^{\leq K}$ is at least as large. However, if A is in general position, then the situation is far better:

▶ **Proposition 5.** *Let $A \subset \mathbb{R}^d$ be a set of n sites in general position, with a constant dimension d. Then* $\text{S-Del}^{\leq K}$ *has size $O(n^{\lfloor \frac{d+1}{2} \rfloor} K^{\lceil \frac{d+1}{2} \rceil})$. In particular,* S-Del *has size $O(n^{d+1})$.*

In brief, the idea of the proof is to bound the number of maximal simplices in $\text{S-Del}^{\leq K}$ using a bound on the number of Voronoi vertices at levels $\leq k$ [18]. The result then follows by observing that the dimension of the complex is a constant that only depends on d. However, this dependence on d is doubly exponential, so the O-notation hides a large factor if d is large. See the full version of this paper for details of the proof.

In contrast to Proposition 5, the number of vertices in Sheehy's subdivision-Čech model [44] grows exponentially with $|A|$, regardless of whether A is in general position.

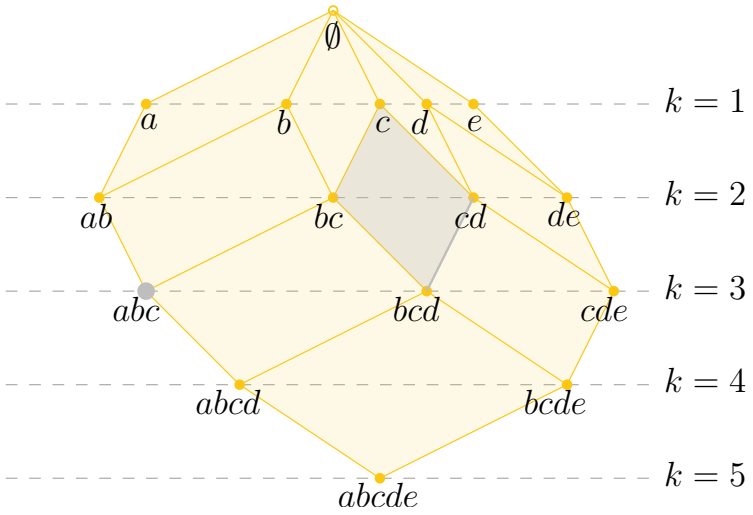


Figure 6 The rhomboid tiling of 5 points on the real line. The highlighted 2-rhomboid ρ defined by $A_{in}(\rho) = \{c\}$ and $A_{on}(\rho) = \{b, d\}$ is the convex hull of the points c, bc, cd, and bcd, simplifying the labels here and, e.g., writing bcd instead of the cell complex associated to $\{b, c, d\}$. The horizontal line at depth k intersects the tiling in the order-k Delaunay mosaic.

4 The rhomboid bifiltration

The rhomboid tiling. Let $A \subseteq \mathbb{R}^d$ be a finite set of sites in general position, and S an arbitrary $(d-1)$-sphere in $\mathbb{R}^d$. Then S yields a decomposition $A = A_{in} \sqcup A_{on} \sqcup A_{out}$ with A_{in} the sites in the interior of S, A_{on} the sites on the sphere, and A_{out} the sites in the exterior of S. We define the *combinatorial rhomboid* of S to be the collection of sets

$$\rho_S := \left\{ A_{in} \cup \tilde{A} \mid \tilde{A} \subseteq A_{on} \right\}. \tag{1}$$

We call elements of ρ_S *combinatorial vertices*, and call

$$\mathrm{Rhomb}(A) = \left\{ \rho_S \mid S \text{ is a sphere in } \mathbb{R}^d \right\} \tag{2}$$

the *(combinatorial) rhomboid tiling* of A. Elements of $\mathrm{Rhomb}(A)$ are called *rhomboids*. Since A is fixed throughout, we write Rhomb instead of $\mathrm{Rhomb}(A)$.

As observed in [26], the combinatorial rhomboid tiling can be geometrically realized as a polyhedral cell complex [35, Def 2.38]. For that, a combinatorial vertex $\{a_1, \ldots, a_k\}$ (where $a_1, \ldots, a_k$ are sites in $\mathbb{R}^d$) is embedded as $(\sum_{i=1}^k a_i, -k)$ in $\mathbb{R}^{d+1}$. We call k the *depth* of the vertex. Embedding a combinatorial rhomboid as the convex hull of its embedded vertices yields an actual rhomboid in $\mathbb{R}^{d+1}$ whose dimension equals the cardinality of A_{on} in the corresponding partition of A. The collection of these rhomboids is the *(geometric) rhomboid tiling* for A. We illustrate the construction in Figure 6. In what follows, we identify vertices and rhomboids with their combinatorial description. In particular, we will use Rhomb both for the combinatorial and the geometric rhomboid tiling.

▶ **Proposition 6** ([41, Proposition 4.8],[22, Section 1.2]). *The total number of cells (of all dimensions) in* Rhomb *is at most* $\frac{2^{d+1}}{(d+1)!}|A+1|^{d+1} \leq 2|A+1|^{d+1}$.

For any rhomboid $\rho \in \mathrm{Rhomb}$, we let r_ρ denote the infimal radius among all spheres S for which $\rho_S = \rho$. It is easily checked that if ρ' is a subset of ρ, then $r_{\rho'} \leq r_\rho$, and therefore, for any $r \geq 0$, the sublevel set $\mathrm{Rhomb}_r = \{\rho \in \mathrm{Rhomb} \mid r_\rho \leq r\}$ is also a polyhedral complex.

Slicing. Next, we slice the rhomboid tiling by cutting every rhomboid along the hyperplanes $\{x \in \mathbb{R}^{d+1} \mid -x_{d+1} = k\}$ with $k = 0, \dots, n$. In this way, a rhomboid decomposes into its intersections with these hyperplanes and with slabs of the form $\{x \in \mathbb{R}^{d+1} \mid k \leq -x_{d+1} \leq k+1\}$. Clearly, the resulting polyhedra again form a polyhedral complex that we call the *sliced rhomboid tiling* S-Rhomb. We refer to its cells as *sliced rhomboids*.

For a sliced rhomboid ρ, we define k_ρ as the minimum depth among its vertices. Moreover, there is a unique (unsliced) rhomboid ρ' of smallest dimension that contains ρ, and we define $r_\rho := r_{\rho'}$. Define

$$\text{S-Rhomb}_{r,k} := \{\rho \in \text{S-Rhomb} \mid r_\rho \leq r, k_\rho \geq k\}$$

and observe that for $r \leq r'$ and $k \geq k'$, we have $\text{S-Rhomb}_{r,k} \subseteq \text{S-Rhomb}_{r',k'}$. Hence, $(\text{S-Rhomb}_{r,k})_{(r,k)\in[0,\infty)\times\mathbb{N}}$ is a bifiltration of combinatorial cell complexes. Again, we will abuse notation and use the symbol S-Rhomb both for the sliced rhomboid tiling and the bifiltration $(\text{S-Rhomb}_{r,k})_{(r,k)\in[0,\infty)\times\mathbb{N}}$. As shown in [26], the restriction of S-Rhomb to cells in the hyperplane $-x_{d+1} = k$ is the order-k Delaunay mosaic, i.e., the geometric dual of the order-k Voronoi diagram.

Comparison of S-Rhomb and S-Del. The following result establishes a close relationship between the bifiltrations S-Rhomb and S-Del:

▶ **Lemma 7.** *For all* $(r, k) \in [0, \infty) \times \mathbb{N}$,

1. *The vertex sets of* $\text{S-Rhomb}_{r,k}$ *and* $\text{S-Del}_{r,k}$ *are equal.*
2. *The vertices of each sliced rhomboid in* $\text{S-Rhomb}_{r,k}$ *span a simplex in* $\text{S-Del}_{r,k}$.
3. *The vertices of each simplex in* $\text{S-Del}_{r,k}$ *are contained in a sliced rhomboid of* $\text{S-Rhomb}_{r,k}$.

Proof. For the first part, note that a set $v = \{a_1, \dots, a_k\}$ of sites is a vertex in S-Rhomb if and only if there is a sphere S that decomposes the sites into $A = A_{in} \sqcup A_{on} \sqcup A_{out}$ such that $A_{in} \subseteq v \subseteq A_{in} \cup A_{on}$. Such a sphere exists if and only if its center has v among its k closest sites, which is equivalent to the condition that the order-k Voronoi region $\text{Vor}(v)$ is not empty. A slightly more careful analysis shows that the vertices of $\text{S-Rhomb}_{r,k}$ and $\text{S-Del}_{r,k}$ are also the same for any choice of r and k.

For the second part, let $v_1, \dots, v_m$ denote the vertices of a sliced rhomboid in $\text{S-Rhomb}_{r,k}$. Let ρ denote the smallest rhomboid of Rhomb containing this sliced rhomboid, and let S denote a sphere of radius $\leq r$ that gives rise to ρ. Let x denote the center of S and $A = A_{in} \sqcup A_{on} \sqcup A_{out}$ be the decomposition with respect to S. Now, each v_i is the union of A_{in} with a subset of A_{on}, and hence, the point x belongs to the Voronoi region of v_i. Since i is arbitrary, it follows that all Voronoi regions intersect, and x has distance $\leq r$ to each v_i, so $v_1, \dots, v_m$ span a simplex in $\text{S-Del}_{r,k}$.

For the third part, consider vertices $v_1, \dots, v_m$ that span a simplex in $\text{S-Del}_{r,k}$. Assume that some v_i has order $k' \geq k$ and that the remaining vertices are of order k' or $k'-1$. Since $v_1, \dots, v_m$ span a simplex, the corresponding higher-order Voronoi regions, intersected with balls of radius r around the involved sites, have non-empty intersection. Choose such a point $x \in \mathbb{R}^d$. Define S as the smallest sphere centered at x that includes k' sites (either on the sphere or in its interior). S induces a partition $A = A_{in} \sqcup A_{on} \sqcup A_{out}$ and a rhomboid ρ. Each vertex v_i of order k' must contain all sites of A_{in}, and some subset of the sites of A_{on}, meaning that v_i is in ρ. Furthermore, at least one site lies on S; otherwise, there would be a smaller sphere. This implies that each vertex v_j of order $k'-1$ also has to contain all sites of A_{in} and some subset of A_{on}. Thus, all vertices lie in ρ, and in particular in its $(k'-1, k')$-slice. Finally, observe that because one of the sites lies on S, the radius of S is the distance of that site to x, which is at most r. ◀

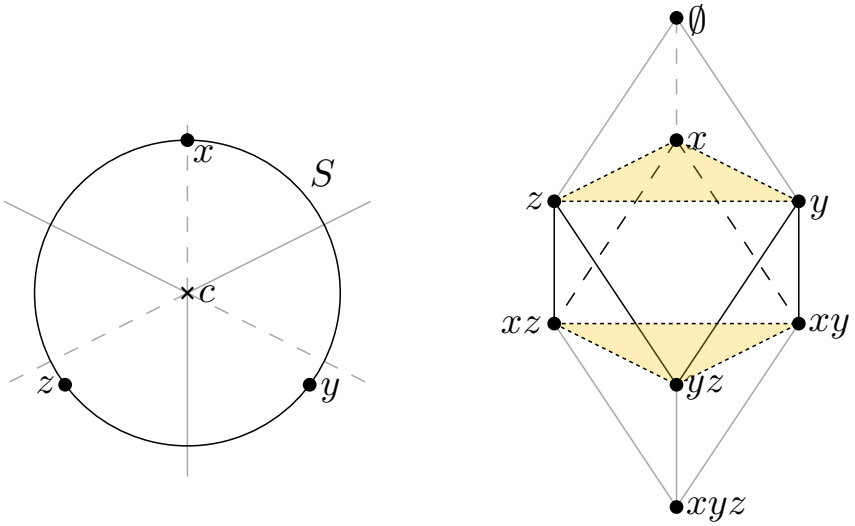


Figure 7 An illustration of the difference between S-Del and S-Rhomb for three points in the plane. The order-1 Voronoi regions of the points $\{x\}$, $\{y\}$, and $\{z\}$ intersect in c, as do the order-2 Voronoi regions of $\{x, y\}$, $\{y, z\}$, and $\{x, z\}$. Consequently, S-Del contains a 5-simplex, but the corresponding cell in S-Rhomb on the same vertex set is a 3-dimensional triangular skew prism.

▶ Remark 8. Parts 1 and 2 of Lemma 7 establish that we have a vertex-preserving injection $\mathcal{J}$ from the cells of S-Rhomb$_{r,k}$ to the simplices of S-Del$_{r,k}$. Moreover, the third part of Lemma 7 implies that $\mathcal{J}$ restricts to a bijection from the maximal cells of S-Rhomb$_{r,k}$ to the maximal simplices of S-Del$_{r,k}$.

We note that $\mathcal{J}$ does not preserve dimension: For ν a cell of S-Rhomb spanned by vertices of cardinality k (i.e., a cell in the order-k Delaunay mosaic), we have $\dim(\nu) \leq d$. But if $d \geq 3$, it can be that $\dim(\mathcal{J}(\nu)) > d$, even when the sites are in general position. For a cell ν of S-Rhomb$_{r,k}$ spanned by vertices of cardinality k and $k+1$, we may have $\dim(\nu) \neq \dim(\mathcal{J}(\nu))$ even for $d = 2$, see Figure 7.

▶ **Theorem 9.** *The bifiltrations* S-Rhomb *and* S-Del *are weakly equivalent.*

Proof. We define good covers of both bifiltrations: First, for S-Rhomb$_{r,k}$, we choose the cover that consists of all its cells. This is a good cover because the cells are convex. The collection of these covers over all choices of r and k yields a good cover $\mathcal{U}$ of the bifiltration S-Rhomb, and the Persistent Nerve Theorem then gives objectwise homotopy equivalences

$$\text{S-Rhomb} \xleftarrow{\simeq} \Delta\mathcal{U} \xrightarrow{\simeq} \text{Nrv}(\mathcal{U}).$$

for some intermediate bifiltration $\Delta\mathcal{U}$.

Moreover, we obtain a cover $\mathcal{V}_{r,k}$ of S-Del$_{r,k}$ whose elements are the simplices spanned by the vertices of the sliced rhomboids in S-Rhomb$_{r,k}$. By the second part of Lemma 7, these cover elements indeed exist. Moreover, in view of Remark 8, every maximal simplex of S-Del$_{r,k}$ is an element of $\mathcal{V}_{r,k}$. We thus obtain a good cover $\mathcal{V}$ of S-Del. Applying the Persistent Nerve Theorem again, we obtain objectwise homotopy equivalences

$$\text{S-Del} \xleftarrow{\simeq} \Delta\mathcal{V} \xrightarrow{\simeq} \text{Nrv}(\mathcal{V}).$$

Finally, $\mathrm{Nrv}(\mathcal{U})$ and $\mathrm{Nrv}(\mathcal{V})$ are isomorphic: The elements of $\mathcal{U}$ and of $\mathcal{V}$ are in 1-to-1 correspondence (with corresponding cover elements having the same vertex set). In both cases, an intersection of cover elements is non-empty if and only if the elements share a vertex, which is determined by their vertex sets. Hence, we have objectwise homotopy equivalences

$$\text{S-Rhomb} \xleftarrow{\simeq} \Delta\mathcal{U} \xrightarrow{\simeq} \mathrm{Nrv}(\mathcal{U}) \xrightarrow{\cong} \mathrm{Nrv}(\mathcal{V}) \xleftarrow{\simeq} \Delta\mathcal{V} \xrightarrow{\simeq} \text{S-Del}. \qquad \blacktriangleleft$$

Unslicing the rhomboid. Next, we define a bifiltration on the (unsliced) rhomboid tiling. Recall that for a rhomboid ρ, we already defined r_ρ as the radius of the smallest sphere that gives rise to that rhomboid. As in the sliced version, we define k_ρ as the minimal depth among the vertices of ρ, and $\mathrm{Rhomb}_{r,k} := \{\rho \mid r_\rho \leq r, k_\rho \geq k\}$. This yields a bifiltration $(\mathrm{Rhomb}_{r,k})_{(r,k)\in[0,\infty)\times\mathbb{N}}$, which we denote by Rhomb.

▶ **Lemma 10.** *The bifiltrations* Rhomb *and* S-Rhomb *are weakly equivalent.*

Proof. For a rhomboid ρ in Rhomb, set $k_{\min}$ as the minimal depth and $k_{\max}$ as the maximal depth among the vertices in ρ; note that $k_\rho = k_{\min}$. For r and k fixed, we call ρ *dangling* if $r_\rho \leq r$ and $k_{\min} < k < k_{\max}$. Note that ρ is not contained in $\mathrm{Rhomb}_{r,k}$ in this case, but some of its slices are contained in $\text{S-Rhomb}_{r,k}$. Moreover, the only difference between $\mathrm{Rhomb}_{r,k}$ and $\text{S-Rhomb}_{r,k}$ are slices of dangling rhomboids in $\text{S-Rhomb}_{r,k}$.

Now, observe that for a dangling rhomboid of maximal dimension, there is a deformation retraction that "pushes in" the slices of the dangling rhomboid, leaving the boundaries of the slices unchanged and removing the entire interior part. Applying this deformation retraction for all maximal-dimensional rhomboids in parallel, and repeating the process for the next lower dimension yields a deformation retraction from $\text{S-Rhomb}_{r,k}$ to $\mathrm{Rhomb}_{r,k}$. This implies that for every choice of r and k, the inclusion $\mathrm{Rhomb}_{r,k} \hookrightarrow \text{S-Rhomb}_{r,k}$ is a homotopy equivalence. Moreover, these inclusions commute with the inclusion maps in Rhomb and S-Rhomb, hence define an objectwise homotopy equivalence. ◀

Combining the previous lemma with Theorem 9 and Theorem 4 yields the following result:

▶ **Theorem 11.** *The bifiltrations* Rhomb *and* Cov *are weakly equivalent.*

▶ Remark 12 (Size of the Rhomboid Bifiltration). In view of Proposition 6, Rhomb has at most $2|A+1|^{d+1} = O(|A|^{d+1})$ cells. One can also bound the size of a truncated version of Rhomb, defined analogously to the truncation of S-Del considered in Proposition 5. Indeed, Rhomb is clearly smaller (in terms of number of cells) than S-Rhomb, and by Remark 8, S-Rhomb is at least as small as S-Del. Moreover, this extends to truncations of these bifiltrations. Thus, the bound of Proposition 5 also holds for truncations of Rhomb.

5 Experiments

The algorithm from [27] computes the rhomboids of the rhomboid tiling and their associated radius values. We extended its implementation RHOMBOIDTILING[1] to compute the sliced and unsliced bifiltrations S-Rhomb and Rhomb and their free implicit representations (FIREP) [40]. The implementation of RHOMBOIDTILING is in C++, using the CGAL library[2] for geometric primitives. The current version accepts only 2- and 3-dimensional inputs, but all steps readily generalize to higher dimensions; adding support for higher-dimensional inputs is a matter of software design rather than algorithm development.

[1] `https://github.com/geoo89/rhomboidtiling`
[2] CGAL, Computational Geometry Algorithms Library, `https://www.cgal.org`

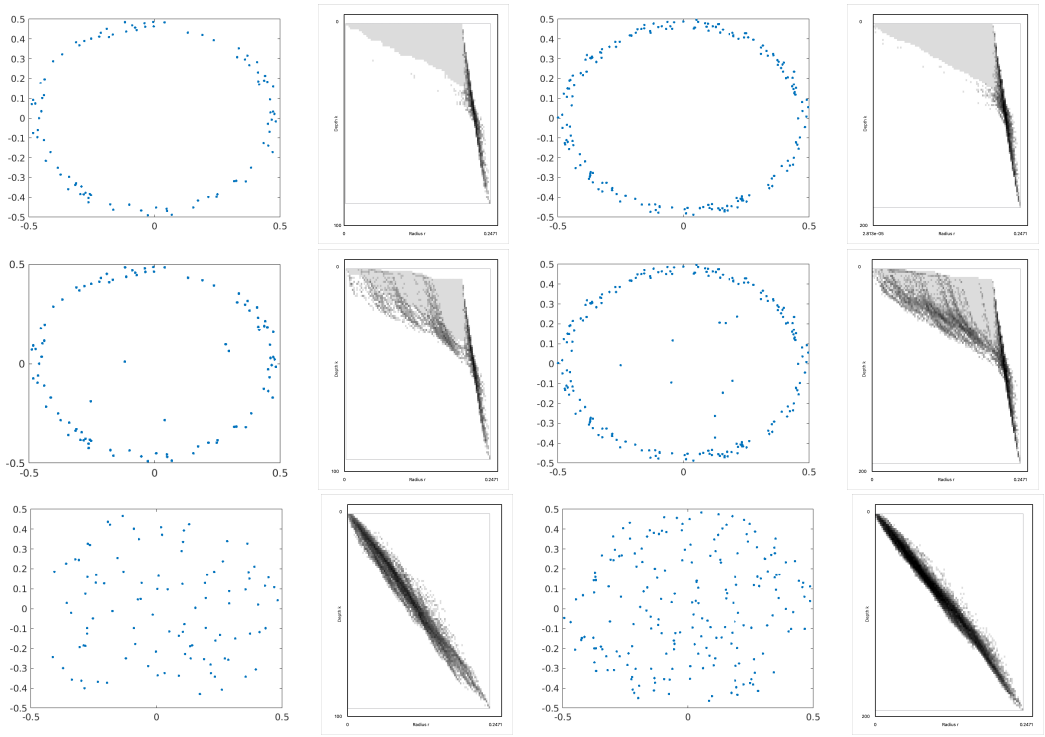


Figure 8 An illustration of the first Hilbert function of the multicover bifiltration, using grayscale shading. The instances are samples of an annulus (top), a noisy annulus (middle), and a disk (bottom). The sample size is 100 in the left column, and 200 in the right column. Darkness of the shading is proportional to the value of the Hilbert function, up to some maximum value, above which the shading is taken to be black; the lightest non-white shade of gray corresponds to a Hilbert function value of 1.

We performed experiments on point sets in $\mathbb{R}^2$ and $\mathbb{R}^3$. We provide a brief summary here; for detailed results, see the full version of this paper. We sampled points uniformly at random from $[0,1]^2$ and $[0,1]^3$, from a disk, from an annulus, and from an annulus with additional noise. We computed the rhomboid bifiltrations $\mathrm{Rhomb}^{\leq K}$ and $\text{S-Rhomb}^{\leq K}$. We then used MPFREE[3] to compute minimal presentations of 2-parameter persistent homology of our bifiltrations.

In one set of experiments, we found that $\mathrm{Rhomb}^{\leq K}$ is up to 47% smaller than $\text{S-Rhomb}^{\leq K}$, and can be computed more than 20% faster. The experiments suggest that the relative performance of $\mathrm{Rhomb}^{\leq K}$ improves with increasing K.

We investigated the size of $\mathrm{Rhomb}^{\leq K}$, varying the sample size and the density parameter. For $d = 2$, our experiments show a clear subquadratic growth of the size of $\mathrm{Rhomb}^{\leq K}$ and its FIREP with respect to increasing K. For $d = 3$, the growth is clearly subcubic. These observations also extend to time complexity. Letting the number of points increase, the size of $\mathrm{Rhomb}^{\leq K}$ and its FIREP shows roughly linear growth for both space dimensions, with a slight superlinear tendency. Again, we observed the same behavior for the computation time.

We conclude this section with a data visualization enabled by the ideas of this paper: For $i \geq 0$, the *i-th Hilbert function* assigns to each parameter $(r,k) \in \mathbb{R} \times \mathbb{N}$ the rank of i-th homology module of $\mathrm{Cov}_{r,k}$ (with coefficients in some fixed field). The Hilbert functions are

[3] `https://bitbucket.org/mkerber/mpfree`

well known to be unstable invariants. Nevertheless, their visualization can give us a feel for how the Lipschitz stability property of the multicover bifiltration established in [6] manifests itself in random data. Figure 8 shows a few examples, plotted using RIVET[4].

6 Conclusion

We have introduced a simplicial model for the multicover bifiltration, as well as a polyhedral model based on the rhomboid tiling [26]. For a data set of size n in $\mathbb{R}^d$ with d constant, the size of both constructions is $O(n^{d+1})$. The size can be further controlled by thresholding the parameter k of the multicover bifiltration. A recently introduced algorithm for computing the rhomboid tiling [27] extends readily to the computation of the polyhedral bifiltration, and we have implemented this extension. In our experimental results, this approach scales well enough to suggest that practical applications could soon be within reach. A natural next step is to begin exploring the use of the multicover bifiltration on real world data.

To obtain our combinatorial models of the multicover bifiltration, we begin with a zigzag of filtrations, and then straighten it out by taking unions of prefixes. Notably, one could in principle compute the persistent homology modules of the multicover bifiltration without straightening out the zigzag, by inverting the isomorphisms on homology induced by the inclusions $\mathrm{Del}_{r,k} \hookrightarrow \widetilde{\mathrm{Del}}_{r,k}$. It seems plausible that this approach could be computationally useful.

We are curious to learn which indecomposables typically arise in the persistent homology modules of multicover bifiltration, and our approach could be used in conjunction with existing algorithms [20, 33] to study this. It would also be interesting to investigate whether there is an interplay between the geometry of a space and the multicover bifiltration of a noisy sample of this space; we wonder if invariants of the bifiltration encode additional information about geometric properties, such as the reach or differentiability.

Our experiments show a significant increase in the size of our models of multicover bifiltration for increasing K. This suggests the need for refinements to our algorithmic approach in order to handle large values of K. Aside from the truncations considered in this paper, there are a couple of promising ways forward: One could construct a coarsened bifiltration where some values of k are skipped. Alternatively, one could make use of the inductive nature of our constructions: for the step from k to $k+1$, one does not need information about the bifiltrations at indices $j < k$. Therefore, one could provide the bifiltration as an output stream without storing it completely in memory. Subsequent algorithmic steps would then have to be implemented as streaming algorithms as well.

References

1 Hirokazu Anai, Frédéric Chazal, Marc Glisse, Yuichi Ike, Hiroya Inakoshi, Raphaël Tinarrage, and Yuhei Umeda. DTM-based filtrations. In *35th International Symposium on Computational Geometry (SoCG 2019)*. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019. doi:10.4230/LIPIcs.SoCG.2019.58.

2 Franz Aurenhammer. Power diagrams: properties, algorithms and applications. *SIAM Journal on Computing*, 16(1):78–96, 1987. doi:10.1137/0216006.

3 Ulrich Bauer, Michael Kerber, Fabian Roll, and Alexander Rolle. A unified view on nerve theorems. *Unpublished manuscript*, 2020.

[4] https://github.com/rivetTDA/

4 Andrew J. Blumberg, Itamar Gal, Michael A. Mandell, and Matthew Pancia. Robust statistics, hypothesis testing, and confidence intervals for persistent homology on metric measure spaces. *Foundations of Computational Mathematics*, 14(4):745–789, 2014. doi:10.1007/s10208-014-9201-4.

5 Andrew J. Blumberg and Michael Lesnick. Universality of the homotopy interleaving distance, 2017. arXiv:1705.01690.

6 Andrew J. Blumberg and Michael Lesnick. Stability of 2-parameter persistent homology, 2020. arXiv:2010.09628.

7 Omer Bobrowski, Sayan Mukherjee, Jonathan E. Taylor, et al. Topological consistency via kernel estimation. *Bernoulli*, 23(1):288–328, 2017. doi:10.3150/15-BEJ744.

8 Mickaël Buchet, Frédéric Chazal, Steve Y. Oudot, and Donald R. Sheehy. Efficient and robust persistent homology for measures. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms, (SODA 2015)*, pages 168–180. SIAM, 2015. doi:10.1137/1.9781611973730.13.

9 Gunnar Carlsson, Tigran Ishkhanov, Vin De Silva, and Afra Zomorodian. On the local behavior of spaces of natural images. *International journal of computer vision*, 76(1):1–12, 2008. doi:10.1007/s11263-007-0056-x.

10 Gunnar Carlsson and Afra Zomorodian. The theory of multidimensional persistence. *Discrete & Computational Geometry*, 42(1):71–93, 2009. doi:10.1007/s00454-009-9176-0.

11 Nicholas J. Cavanna, Kirk P. Gardner, and Donald R. Sheehy. When and why the topological coverage criterion works. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms, (SODA 2017)*, 2017. doi:10.1137/1.9781611974782.177.

12 Nicholas J. Cavanna, Mahmoodreza Jahanseir, and Donald R. Sheehy. A geometric perspective on sparse filtrations. In *Proceedings of the 27th Canadian Conference on Computational Geometry (CCCG 2015)*, 2015.

13 Andrea Cerri, Barbara Di Fabio, Massimo Ferri, Patrizio Frosini, and Claudia Landi. Betti numbers in multidimensional persistent homology are stable functions. *Mathematical Methods in the Applied Sciences*, 36(12):1543–1557, 2013. doi:10.1002/mma.2704.

14 Frédéric Chazal, Leonidas J. Guibas, Steve Y. Oudot, and Primoz Skraba. Scalar field analysis over point cloud data. *Discrete & Computational Geometry*, 46(4):743–775, 2011. doi:10.1007/s00454-011-9360-x.

15 Frédéric Chazal, Leonidas J. Guibas, Steve Y. Oudot, and Primoz Skraba. Persistence-based clustering in Riemannian manifolds. *Journal of the ACM*, 60(6), 2013. doi:10.1145/2535927.

16 Frédéric Chazal and Steve Y. Oudot. Towards persistence-based reconstruction in Euclidean spaces. In *24th International Symposium on Computational Geometry (SoCG 2008)*, page 232–241. Association for Computing Machinery (ACM), 2008. doi:10.1145/1377676.1377719.

17 Frédéric Chazal, David Cohen-Steiner, and Quentin Mérigot. Geometric inference for probability measures. *Foundations of Computational Mathematics*, 11:733–751, December 2011. doi:10.1007/s10208-011-9098-0.

18 Kenneth L. Clarkson and Peter W. Shor. Applications of random sampling in computational geometry, II. *Discrete & Computational Geometry*, 4(5):387–421, 1989. doi:10.1007/BF02187740.

19 David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Stability of persistence diagrams. *Discrete & Computational Geometry*, 37(1):103–120, 2007. doi:10.1007/s00454-006-1276-5.

20 Tamal K. Dey and Cheng Xin. Generalized persistence algorithm for decomposing multi-parameter persistence modules, 2020. arXiv:1904.03766.

21 William G. Dwyer and Jan Spalinski. Homotopy theories and model categories. *Handbook of algebraic topology*, 73:126, 1995. doi:10.1016/B978-044481779-2/50003-1.

22 Herbert Edelsbrunner. *Algorithms in combinatorial geometry*. Springer-Verlag, 1987. doi:10.1007/978-3-642-61568-9.

23 Herbert Edelsbrunner. The union of balls and its dual shape. *Discrete & Computational Geometry*, 13(3):415–440, 1995. doi:10.1007/BF02574053.

24 Herbert Edelsbrunner. Shape reconstruction with Delaunay complex. In *LATIN'98: Theoretical Informatics*, pages 119–132. Springer Berlin Heidelberg, 1998. doi:10.1007/BFb0054315.

25 Herbert Edelsbrunner and John Harer. *Computational topology: an introduction*. American Mathematical Society, 2010. doi:10.1007/978-3-540-33259-6_7.

26 Herbert Edelsbrunner and Georg Osang. The multi-cover persistence of Euclidean balls. In *34th International Symposium on Computational Geometry (SoCG 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018. doi:10.4230/LIPIcs.SoCG.2018.34.

27 Herbert Edelsbrunner and Georg Osang. A simple algorithm for computing higher order Delaunay mosaics and α-shapes, 2020. arXiv:2011.03617.

28 Herbert Edelsbrunner and Raimund Seidel. Voronoi diagrams and arrangements. *Discrete & Computational Geometry*, 1(1):25–44, 1986. doi:10.1007/BF02187681.

29 Leonidas Guibas, Dmitriy Morozov, and Quentin Mérigot. Witnessed k-distance. *Discrete & Computational Geometry*, 49(1):22–45, 2013. doi:10.1007/s00454-012-9465-x.

30 Heather A. Harrington, Nina Otter, Hal Schenck, and Ulrike Tillmann. Stratifying multiparameter persistent homology, 2017. arXiv:1708.07390.

31 Allen Hatcher. *Algebraic Topology*. Cambridge University Press, 2005.

32 Philip S. Hirschhorn. *Model categories and their localizations*, volume 99. American Mathematical Society, 2009.

33 Derek F. Holt. The Meataxe as a tool in computational group theory. *London Mathematical Society Lecture Note Series*, pages 74–81, 1998.

34 Michael Kerber and Alexander Rolle. Fast minimal presentations of bi-graded persistence modules. In *2021 Proceedings of the Symposium on Algorithm Engineering and Experiments (ALENEX)*, pages 207–220. SIAM, 2021. doi:10.1137/1.9781611976472.16.

35 Dmitry Kozlov. *Combinatorial Algebraic Topology*. Springer, 2008. doi:10.1007/978-3-540-71962-5.

36 Dmitry Krasnoshchekov and Valentin Polishchuk. Order-k alpha-hulls and alpha-shapes. *Information Processing Letters*, 114(1-2):76–83, 2014. doi:10.1016/j.ipl.2013.07.023.

37 Edoardo Lanari and Luis N. Scoccola. Rectification of interleavings and a persistent Whitehead theorem, 2020. arXiv:2010.05378.

38 Jean Leray. Sur la forme des espaces topologiques et sur les points fixes des representations. *Journal de Mathématiques Pures et Appliquées*, 24, January 1945.

39 Michael Lesnick and Matthew Wright. Interactive visualization of 2-D persistence modules, 2015. arXiv:1512.00180.

40 Michael Lesnick and Matthew Wright. Computing minimal presentations and Betti numbers of 2-parameter persistent homology, 2019. arXiv:1902.05708.

41 Georg F. Osang. *Multi-cover persistence and Delaunay mosaics*. PhD thesis, IST Austria, 2021. doi:10.15479/AT:ISTA:9056.

42 Jeff M. Phillips, Bei Wang, and Yan Zheng. Geometric inference on kernel density estimates. In *31st International Symposium on Computational Geometry (SoCG 2015)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2015. doi:10.4230/LIPIcs.SOCG.2015.857.

43 Luis N. Scoccola. *Locally Persistent Categories And Metric Properties Of Interleaving Distances*. PhD thesis, The University of Western Ontario, 2020. URL: https://ir.lib.uwo.ca/etd/7119/.

44 Donald R. Sheehy. A multicover nerve for geometric inference. In *Proceedings of the 24th Canadian Conference in Computational Geometry (CCCG 2012)*, 2012.

45 Donald R. Sheehy. Linear-size approximations to the Vietoris–Rips filtration. *Discrete & Computational Geometry*, 49(4):778–796, 2013. doi:10.1007/s00454-013-9513-1.

46 Donald R. Sheehy. A sparse delaunay filtration, 2020. arXiv:2012.01947.

47 Oliver Vipond. Multiparameter persistence landscapes. *Journal of Machine Learning Research*, 21(61):1–38, 2020. URL: http://jmlr.org/papers/v21/19-054.html.

48 Georgy Voronoi. Recherches sur les paralléloèdres primitives. *Journal für die reine und angewandte Mathematik*, 134:198–287, 1908.

49 Afra Zomorodian and Gunnar Carlsson. Computing persistent homology. *Discrete and Computational Geometry*, 33(2):249–274, 2005. doi:10.1007/s00454-004-1146-y.

Escaping the Curse of Spatial Partitioning: Matchings with Low Crossing Numbers and Their Applications

Mónika Csikós
Université Gustave Eiffel, LIGM, Equipe A3SI, ESIEE Paris,
Cité Descartes 2 boulevard Blaise Pascal, 93162 Noisy-le-Grand Cedex, France

Nabil H. Mustafa
Université Gustave Eiffel, LIGM, Equipe A3SI, ESIEE Paris,
Cité Descartes 2 boulevard Blaise Pascal, 93162 Noisy-le-Grand Cedex, France

Abstract

Given a set system $(X, \mathcal{S})$, constructing a matching of X with low crossing number is a key tool in combinatorics and algorithms. In this paper we present a new sampling-based algorithm which is applicable to finite set systems. Let $n = |X|$, $m = |\mathcal{S}|$ and assume that X has a perfect matching M such that any set in $\mathcal{S}$ crosses at most $\kappa = \Theta(n^\gamma)$ edges of M. In the case $\gamma = 1 - 1/d$, our algorithm computes a perfect matching of X with expected crossing number at most 10κ, in expected time $\tilde{O}\left(n^{2+2/d} + mn^{2/d}\right)$.

As an immediate consequence, we get improved bounds for constructing low-crossing matchings for a slew of both abstract and geometric problems, including many basic geometric set systems (e.g., balls in $\mathbb{R}^d$). This further implies improved algorithms for many well-studied problems such as construction of ϵ-approximations. Our work is related to two earlier themes: the work of Varadarajan (STOC '10) / Chan et al. (SODA '12) that avoids spatial partitionings for constructing ϵ-nets, and of Chan (DCG '12) that gives an optimal algorithm for matchings with respect to hyperplanes in $\mathbb{R}^d$.

Another major advantage of our method is its simplicity. An implementation of a variant of our algorithm in `C++` is available on Github; it is approximately 200 lines of basic code without any non-trivial data-structure. Since the start of the study of matchings with low-crossing numbers with respect to half-spaces in the 1980s, this is the first implementation made possible for dimensions larger than 2.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Matchings, crossing numbers, approximations

Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.28

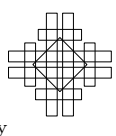
Supplementary Material *Software (Source Code)*:
`https://github.com/csikosm/LowCrossingMatchings`
archived at `swh:1:dir:527290e997b57b959e879363b26dd9218906f3a8`

Funding The work of the authors has been supported by the grants ANR ADDS (ANR-19-CE48-0005) and ANR SAGA (JCJC-14-CE25-0016-01).

1 Introduction

Given a set system $(X, \mathcal{S})$, we say that a set $S \in \mathcal{S}$ *crosses* a pair $\{x, y\} \subseteq X$ iff $|S \cap \{x, y\}| = 1$. Define the *crossing number* of a perfect matching (resp. a spanning tree) G of X with respect to $\mathcal{S}$ as the maximum number of edges of G crossed by any $S \in \mathcal{S}$. The focus of this paper is on constructing perfect matchings of X with low crossing numbers with respect to $\mathcal{S}$.

Matchings with low crossing numbers were originally introduced by Welzl [31, 32] for the special case where X is a set of points in $\mathbb{R}^d$ and $\mathcal{S}$ is induced on X by half-spaces. His result was then generalized by Chazelle and Welzl [10] to a broader class of set systems, which together with an improvement due to Haussler [20], gives the following general theorem.



▶ **Theorem A.** *Let $(X, \mathcal{S})$ be a set system with $n = |X|$, and dual shatter function*[1] $\pi^*_{\mathcal{S}}(k) = O(k^d)$. *Then there exists a perfect matching of X with crossing number $O\left(n^{1-1/d}\right)$.*

Previous constructions. Let $(X, \mathcal{S})$ be a set system with $n = |X|$, $m = |\mathcal{S}|$, and let κ denote the smallest integer such that X has a perfect matching (resp. spanning tree) with crossing number κ with respect to $\mathcal{S}$. We review previous constructions in two separate settings.

Abstract set systems. The original method of Welzl [31, 32, 10] builds a perfect matching using the multiplicative weight update (MWU) method. Briefly, the algorithm maintains a weight function π on $\mathcal{S}$, with initial weights set to 1. It selects edges iteratively, always choosing an edge that is guaranteed to be crossed by sets of low total weight in π; it then updates π based on the chosen edge. The algorithmic bottleneck is in finding such an edge: for an abstract set system without additional structure, this takes $O(n^2 m)$ time for each of the $n/2$ iterations.

Another approach for the abstract case was proposed by Har-Peled [19] (see also [14]). His result implies that if $\kappa = \Theta(n^\gamma)$ for some $\gamma \in [1/\log n, 1]$, then a spanning tree of crossing number $O(\kappa/\gamma)$ can be found by solving an LP on $\binom{n}{2}$ variables and $m + n$ constraints. Combining this with an efficient approximate LP solver (e.g., [11]) leads to a randomized $\tilde{O}(mn^2)$ time algorithm. The approximation factor can be further improved using a general framework of rounding fractional solutions of minimax integer programs with matroid constraints. This method gives a randomized algorithm that constructs a spanning tree with expected crossing number at most $\kappa + O(\sqrt{\kappa \log m})$ in time $\tilde{O}(mn^4 + n^8)$ [12].

Geometric set systems. Now we turn to the case where X is a set of n points in $\mathbb{R}^d$ and $\mathcal{S}$ consists of subsets of X that are induced by geometric objects. In this setting, improved bounds are made possible using spatial partitioning. The current-best algorithms for geometric set systems induced by half-spaces recursively construct simplicial partitions, stored in a hierarchical structure called the partition tree, which then at its base level gives a matching with low crossing number. This approach is used in the breakthrough result of Chan [8] who gave an $O(n \log n)$ time algorithm to build partition trees with respect to half-spaces in $\mathbb{R}^d$, which then implies the same for computing matchings with crossing number $O(n^{1-1/d})$.

While the use of cuttings – and more generally, spatial partitioning – gives $o(mn^2)$ running times, progress remains blocked in several ways:

a) Simplicial partitions only exist in certain geometric settings. Indeed, as shown by Alon et al.[6], they do not always exist in settings satisfying the requirements of Theorem A (e.g., the projective plane). Furthermore, spatial partitioning is not possible when dealing with abstract set systems such as those arising in graph theory or learning theory.

b) Optimal bounds for constructing simplicial partitions are only known for the case of half-spaces; this is one of the main problems left open by Chan [8]. Despite a series of research for semi-algebraic set systems (using linearization, cuttings, and more recently, polynomial partitioning [3]), the bounds are still sub-optimal for polynomials of degree larger than 2, with exponential dependence on the dimension.

c) There are large constants in the asymptotic notation depending on the dimension d both in the running time as well as the crossing number bounds, due to the use of cuttings (see [13]). For instance, in Chan's algorithm the constants are quite large – Theorem 3.2 [8]

[1] The *dual shatter function* $\pi^*_{\mathcal{S}}$ of $(X, \mathcal{S})$ is defined as follows. For any $k \leq |\mathcal{S}|$, $\pi^*_{\mathcal{S}}(k)$ is the maximum number of equivalence classes on X defined by a k-element subfamily $\mathcal{R} \subseteq \mathcal{S}$, where $x, y \in X$ are equivalent with respect to $\mathcal{R}$ if x belongs to the same sets of $\mathcal{R}$ as y.

requires $\delta \leq \frac{1}{d^2}$, $b = 22$ (see [21]), which then implies that it constructs a spanning tree with a guaranteed crossing number no better than $12 \cdot 22 \cdot d^4 n^{1-1/d}$; this is at least $20000 \cdot n^{1-1/d}$ even for $d = 3$. Furthermore, the actual construction running time is at least $264 \cdot d^2 n \log n$, not counting the typically large constants in the several complex data structures that the algorithm needs (simplex range searching in $\mathbb{R}^d$ with dynamic insertion; see [21] for a discussion of its practical aspects in $\mathbb{R}^2$).

d) Practical implementation of spatial partitioning in $\mathbb{R}^d$, $d > 2$, even cuttings for hyperplanes, remains an open problem in geometric computing. Cuttings have been implemented in the planar case [18], which have then been used recently for computing ε-approximations w.r.t. half-spaces in $\mathbb{R}^2$ [21]. In particular, for $d > 2$, we know of no implementations for low-crossing matchings.

Recently there have been algorithms proposed for ε-nets and ε-approximations that avoid spatial partitioning [29, 9, 27, 26]. Our work can be considered another step along this theme.

2 Our Results

We state our main result assuming that we have access to a membership Oracle of $(X, \mathcal{S})$, which for a given element $x \in X$ and a set $S \in \mathcal{S}$ returns whether $x \in S$. Our main result is the following.

▶ **Theorem 1.** *Let $(X, \mathcal{S})$ be a set system, $n = |X|$, $m = |\mathcal{S}|$ with $m \geq n$. Let $a > 0$, b and $\gamma \in [1/\log n, 1]$ be constants such that any $Y \subseteq X$ has a perfect matching with crossing number at most $a|Y|^\gamma + b$. Then* BuildMatching$\big((X, \mathcal{S}), a, b, \gamma\big)$ *computes a perfect matching of X with expected crossing number at most $\frac{5a}{\gamma} n^\gamma + (3b + 8 \ln m) \log n$, and with an expected $O\left(\min\left\{n^{4-2\gamma} \ln n + mn^{2-2\gamma} \ln m \ln n,\, n^3 + mn\right\}\right)$ calls to the membership Oracle of $(X, \mathcal{S})$.*

Remarks:

- The algorithm BuildMatching is presented in Section 3.
- Our method can easily be modified to construct a spanning tree or a spanning path with the same guarantees up to a constant factor.

Now we give a list of consequences of Theorem 1, divided into three topics. All stated crossing number and running time bounds are in expectation.

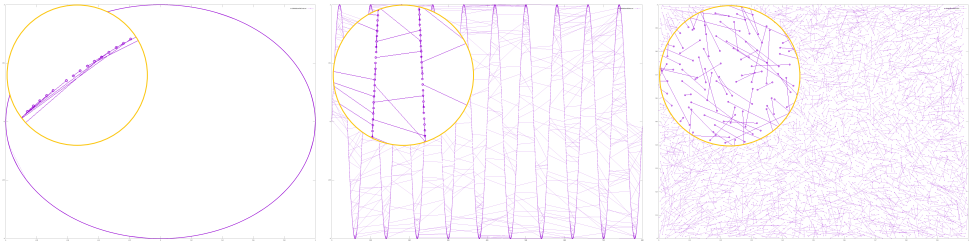
1. **Low-crossing matchings.** Our results improve upon several previous constructions, see Table 1 (the precise guarantees and their proofs are presented in Section 4). For abstract set systems with dual shatter function $\pi^*_{\mathcal{S}}(k) = O(k^d)$, we improve the running time from $\tilde{O}\left(mn^2\right)$ to $\tilde{O}\left(mn^{2/d} + n^{2+2/d}\right)$. This further implies a sub-cubic time construction for matchings with asymptotically-optimal crossing number with respect to balls in $\mathbb{R}^d$ for $d \geq 3$. For set systems induced by semialgebraic sets in $\mathbb{R}^d$ (each set defined by at most s polynomial inequalities of degree at most Δ), we significantly improve the crossing number guarantee by removing the exponential dependence on d. However in contrast to the previous best algorithm for this setup [3], our running time depends on m.
   Importantly, our method does not use spatial partitioning, which makes it possible to handle abstract set-systems, and geometric set systems in $\mathbb{R}^d$ (not only in $\mathbb{R}^2$) without additional complications.
2. **Practical aspects.** Our algorithm consists of $\frac{n}{2}$ iterations, where each iteration adjusts the weight of a random subset of $\binom{X}{2}$ and $\mathcal{S}$ and adds a randomly picked edge to the matching. The only black-box needed is the membership Oracle that returns for a given

Table 1 Summary of our results for set systems $(X, \mathcal{S})$ with $n = |X|$, $m = |\mathcal{S}|$, $n \leq m$, and $d \geq 2$. We use the notation $\pi^*_{\mathcal{S}}(\cdot)$ for the dual shatter function of $(X, \mathcal{S})$, $\mathcal{H}_d$ for half-spaces in $\mathbb{R}^d$, $\mathcal{B}_d$ for balls in $\mathbb{R}^d$, and $\Gamma_{d,\Delta,s}$ for semialgebraic ranges in $\mathbb{R}^d$ described by at most s equations of degree at most Δ (see Sec. 4).

	Matchings / Spanning trees			
	Our method		Previous-best	
Set system	Crossing number	time	Crossing number	time
arbitrary with $\pi^*_{\mathcal{S}}(k) \leq ck^d$	$\left(\frac{5c^{1/d}d}{d-1} + o(1)\right) n^{1-1/d}$	$\boldsymbol{\tilde{O}\left(mn^{2/d}+n^{2+2/d}\right)}$ (Corollary 12)	$O\left(n^{1-1/d}\right)$	$\tilde{O}(mn^2)$ [19, 11]
geometric induced by $\mathcal{B}_d$	$\left(6d^2 + o(d^2)\right) \cdot n^{1-1/d}$	$\boldsymbol{\tilde{O}\left(dn^{2+2/d}\right)}$ (Corollary 18)	$O\left(n^{1-1/d}\right)$	$O\left(n^{3+1/d}\right)$ [19, 11]
geometric induced by $\Gamma_{d,\Delta,s}$	$\frac{20e\Delta sd}{d-1} n^{1-1/d} + o\left(n^{1-1/d}\right)$	$\boldsymbol{\tilde{O}\left(s\Delta^d\left(mn^{2/d}+n^{2+2/d}\right)\right)}$ (Corollary 14)	$O\left(10^d s\Delta n^{1-1/d}\right)$ $O\left(\Delta s n^{1-1/d}\right)$	$O\left(n^{O(d^3)}\right)$ [3] $\tilde{O}(s\Delta^d mn^2)$ [19, 11]
geometric induced by $\mathcal{H}_d$	$\left(6d^2 + o(d^2)\right) \cdot n^{1-1/d}$	$\tilde{O}\left(dn^{2+2/d}\right)$ (Corollary 16)	$\geq 264d^4 n^{1-1/d}$	$\tilde{O}(d^2 n)$ [8]

$x \in X$ and $S \in \mathcal{S}$, if $x \in S$. The time complexity of this operation depends on the precise way $(X, \mathcal{S})$ is given; typically this is independent of $|X|$ and $|\mathcal{S}|$ (using indexing, hashing). A preliminary multi-threaded implementation of a variant of our algorithm in `C++` for set systems induced on points by half-spaces in $\mathbb{R}^d$ is available on Github. It is approximately 200 lines of basic code without any non-trivial data-structures, being the first such implementation for $d > 2$.

The figures below show the matchings with respect to half-planes returned by our algorithm for $5,000$ points in $\mathbb{R}^2$ uniformly placed on a circle (in 17.39s), sine curve (in 17.17s), and randomly perturbed in a uniform grid (in 17.41s), each with a zoomed-in region. We find it surprising that our method, that is based only on random sampling, gives a matching that adapts so well to each specific instance.



This makes progress towards the goals expressed at the end of the survey on range searching and its applications [1]: "...an interesting open question is to develop simple data structures that work well under some assumptions on input points and query ranges".

3. **Discrepancy and approximations.** By plugging in various upper-bounds on crossing numbers given by Theorem 1 and using techniques in Matoušek et al.[25], we immediately get improved construction bounds for discrepancy and ε-approximations. In particular, if d is a constant such that $(X, \mathcal{S})$ has dual shatter function $\pi^*_{\mathcal{S}}(k) = O(k^d)$, then we improve the running time of computing colorings with expected discrepancy $O\left(\sqrt{n^{1-1/d} \ln m}\right)$ from $O(mn^2)$ to $\tilde{O}(mn^{2/d}+n^{2+2/d})$. Moreover if in addition, $(X, \mathcal{S})$ has VC dimension bounded by a constant $D \geq 2$, then our method can be used to compute an ε-approximations of

expected size $\tilde{O}\left(\left(\frac{D}{\varepsilon^2}\right)^{\frac{d}{d+1}}\right)$ in expected time $\tilde{O}\left(n+\left(D\left(\frac{D}{\varepsilon^2}\right)^{D+2/d}\right)\right)$, improving upon the previous-best time $O\left(n+\left(\frac{D}{\varepsilon^2}\right)^{D+2}\right)$. As these are standard applications of matchings with low crossing number, the proofs are omitted (see the survey [28]).

Organization. In Section 3, we describe our algorithm and prove Theorem 1. In Section 4, we show how Theorem 1 implies the bounds stated in Table 1. In Section 5, we present our experiments. In Section 6, we give an application in learning theory.

3 Proof of Theorem 1

The proof rests on the following three key ideas:

1. We replace the bottleneck algorithmic step of finding a light edge in the multiplicative weights update technique by simply sampling an edge according to a carefully maintained distribution. In particular, we maintain weights not only on the sets in $\mathcal{S}$, but also on $\binom{X}{2}$. At each iteration we sample an edge e and a set S according to the current weights. Then we add e to our matching and update the weights by *doubling* the weight of each set that crosses e and *halving* the weight of each edge that is crossed by S. The idea of maintaining "primal-dual" weights has been used earlier to approximately solve matrix games [17] and in geometric optimization [4].
2. In our case, the process is more elaborate as we are constructing a matching M at the same time as reweighing. Therefore, at the end of each iteration, as we add e to M, we are forced to set the weights of e and all edges adjacent to e to 0. This breaks down the reweighing scheme, as the removal of the edges amplifies the error introduced in later iterations and thus our maintained weights degrade over time. However, we prove that restarting the algorithm by "resetting" all the weights a logarithmic number of times suffices to ensure the required low crossing numbers.
3. This still does not get us to our goal as updating the weights of *all* edges and sets crossing the randomly picked set and edge would be too expensive. Instead, we show that if $\gamma \geq 1/2$, then updating the weights of a *uniform* sample of $\tilde{O}\left(n^{3-2\gamma}\right)$ edges and $\tilde{O}(mn^{1-2\gamma})$ sets at each iteration is sufficient for our purposes.

The main algorithm BuildMatching is given below, followed by the presentation of the subroutine MatchHalf.

Algorithm 1 BuildMatching$\big((X,\mathcal{S}),a,b,\gamma\big)$.

$M \leftarrow \emptyset$
while $|X| \geq 4$ **do**
  $M' \leftarrow$ MatchHalf $((X,\mathcal{S}), a|X|^{\gamma}+b)$ // M' covers $|X|/2$ elements
  $M \leftarrow M \cup M'$
  $X \leftarrow X \setminus \text{vertices}(M')$ // remove elements covered by M'
$M \leftarrow M \cup \{\text{edge connecting the remaining two elements of } X\}$
return M

Algorithm 2 MatchHalf$\big((X,\mathcal{S}),\kappa\big)$.

$\omega_1(e) \leftarrow 1, \quad \pi_1(S) \leftarrow 1 \qquad \forall e \in E,\ S \in \mathcal{S}$ // E denotes $\binom{X}{2}$
$\mathbf{p} \leftarrow \min\{106 \cdot |X|/\kappa^2 \cdot \ln(|E| \cdot |X|/4),\ 1\}$
$\mathbf{q} \leftarrow \min\{39 \cdot |X|/\kappa^2 \cdot \ln(|\mathcal{S}| \cdot |X|/4),\ 1\}$
for $i = 1, \ldots, |X|/4$ **do**
  $\omega_i(E) \leftarrow \sum_{e\in E} \omega_i(e)$
  $\pi_i(\mathcal{S}) \leftarrow \sum_{S\in\mathcal{S}} \pi_i(S)$
  choose $e_i \sim \omega_i$ // $\mathbb{P}[e_i = e] = \frac{\omega_i(e)}{\omega_i(E)} \quad \forall e \in E$
  choose $S_i \sim \pi_i$ // $\mathbb{P}[S_i = S] = \frac{\pi_i(S)}{\pi_i(\mathcal{S})} \quad \forall S \in \mathcal{S}$
  $E_i \leftarrow$ sample from E with probability $\mathbf{p}$ // $\mathbb{P}[e \in E_i] = \mathbf{p} \quad \forall e \in E$
  $\mathcal{S}_i \leftarrow$ sample from $\mathcal{S}$ with probability $\mathbf{q}$ // $\mathbb{P}[S \in \mathcal{S}_i] = \mathbf{q} \quad \forall S \in \mathcal{S}$
  // $\mathrm{I}(e,S) = 1$ if e crosses S, $\mathrm{I}(e,S) = 0$ otherwise
  for $e \in E_i$ **do**
    $\omega_{i+1}(e) \leftarrow \omega_i(e)\big(1 - \frac{1}{2}\mathrm{I}(e, S_i)\big)$ // halve weight if S_i crosses e
  for $S \in \mathcal{S}_i$ **do**
    $\pi_{i+1}(S) \leftarrow \pi_i(S)\big(1 + \mathrm{I}(e_i, S)\big)$ // double weight if S crosses e_i
  set the weight in ω_{i+1} of e_i and of each edge adjacent to e_i to zero
return $\{e_1, \ldots, e_{|X|/4}\}$

Proof of Theorem 1. Later, we will prove the following statement for MatchHalf.

▶ **Theorem 2.** *Let $(X,\mathcal{S})$ be a set system, $n = |X|$, $m = |\mathcal{S}|$ with $m \geq n$, and let κ be such that any $Y \subseteq X$ of size $|Y| = n/2$ has a perfect matching of crossing number at most κ with respect to $\mathcal{S}$. Then MatchHalf$\big((X,\mathcal{S}),\kappa\big)$ returns a matching of size $n/4$ with expected crossing number at most $5\kappa/2 + 8\ln m$, with expected $O\left(\min\left\{n^4\ln(n)/\kappa^2 + mn^2\ln(m)/\kappa^2,\ n^3 + mn\right\}\right)$ calls to the membership Oracle of $(X,\mathcal{S})$.*

The proof of Theorem 1 follows by applying Theorem 2 to each of the $\log n$ calls of MatchHalf. We get that the expected crossing number of the matching returned by BuildMatching$\big((X,\mathcal{S}),a,b,\gamma\big)$ is at most

$$\sum_{i=1}^{\log n}\left[\frac{5a}{2}\left(\frac{n}{2^i}\right)^{\gamma} + \frac{5b}{2} + 8\ln m\right] < \left(\frac{5b}{2} + 8\ln m\right)\log n + \frac{5an^{\gamma}}{2}\sum_{i=1}^{\infty}\left(\frac{1}{2^{\gamma}}\right)^{i} < (3b + 8\ln m)\log n + \frac{5a}{\gamma}n^{\gamma}.$$

If $1/2 < \gamma \leq 1$, then the overall expected number of calls to the membership Oracle is

$$\sum_{i=0}^{\log n} O\left(\frac{\left(\frac{n}{2^i}\right)^4 \ln\left(\frac{n}{2^i}\right)}{\left(a\left(\frac{n}{2^i}\right)^{\gamma} + b\right)^2} + \frac{m\left(\frac{n}{2^i}\right)^2 \ln m}{\left(a\left(\frac{n}{2^i}\right)^{\gamma} + b\right)^2}\right) = O\left(n^{4-2\gamma}\ln n + mn^{2-2\gamma}\ln m\log n\right),$$

and if $\gamma \leq 1/2$, the overall expected number of calls to the membership Oracle is

$$\sum_{i=0}^{\log n} O\left(\left(\frac{n}{2^i}\right)^3 + m\left(\frac{n}{2^i}\right)\right) = O\left(n^3 + mn\right). \qquad \blacktriangleleft$$

Proof of Theorem 2. The proof relies on the following technical lemma, whose proof is presented later in this section. For an edge e and a set S, we define $\mathrm{I}(e,S)$ to be 1 if S crosses e and 0 otherwise.

▶ **Lemma 3** (Main Lemma). *Let $\tilde{E}$ denote the set of edges that have non-zero weight when* MatchHalf$\big((X,\mathcal{S}),\kappa\big)$ *terminates. Then*

$$\mathbb{E}\left[\max_{S\in\mathcal{S}}\sum_{i=1}^{n/4}\mathrm{I}(e_i,S)\right] \le 2\cdot\mathbb{E}\left[\min_{e\in\tilde{E}}\sum_{i=1}^{n/4}\mathrm{I}(e,S_i)\right]+\frac{\kappa}{2}+\frac{2\ln|E|+\ln|\mathcal{S}|}{\ln 2}. \tag{1}$$

The left-hand side of Equation (1) is precisely the expected crossing number of the edges returned by MatchHalf. We use the following "short-edge" lemma.

▶ **Lemma 4.** *Let $(Y,\mathcal{R})$ be a set system and κ be such that Y has a perfect matching with crossing number at most κ with respect to $\mathcal{R}$. Then there is an edge spanned by the points of Y that is crossed by at most $\frac{2|\mathcal{R}|\kappa}{|Y|}$ sets of $\mathcal{R}$.*

Proof. Let M be a matching of Y such that any set of $\mathcal{R}$ crosses at most κ edges of M. Then there are at most $|\mathcal{R}|\cdot\kappa$ crossings between the edges of M and sets in $\mathcal{R}$. By the pigeonhole principle, there is an edge in M that is crossed by at most

$$\frac{|\mathcal{R}|\cdot\kappa}{|M|}=\frac{|\mathcal{R}|\cdot\kappa}{|Y|/2}=\frac{2|\mathcal{R}|\kappa}{|Y|}$$

sets of $\mathcal{R}$. ◀

Let $\tilde{X}\subset X$ denote the set of points that are not covered by the edges $\{e_1,\dots,e_{n/4}\}$ returned by MatchHalf$\big((X,\mathcal{S}),\kappa\big)$. Note that $\tilde{E}=\binom{\tilde{X}}{2}$. Applying Lemma 4 to $Y=\tilde{X}$ and $\mathcal{R}=\{S_1,\dots,S_{n/4}\}$, we get that there is an edge $e\in\tilde{E}$ that satisfies

$$\sum_{i=1}^{n/4}\mathrm{I}(e,S_i)\le\frac{2\cdot|\mathcal{R}|\cdot\kappa}{|\tilde{X}|}=\frac{2\cdot n/4\cdot\kappa}{n/2}=\kappa. \tag{2}$$

Thus, by using the Main Lemma, the expected crossing number of the edges $\{e_1,\dots,e_{n/4}\}$ with respect to $\mathcal{S}$ can be bounded as

$$\mathbb{E}\left[\max_{S\in\mathcal{S}}\sum_{i=1}^{n/4}\mathrm{I}(e_i,S)\right] \le 2\cdot\mathbb{E}\left[\min_{e\in\tilde{E}}\sum_{i=1}^{n/4}\mathrm{I}(e,S_i)\right]+\frac{\kappa}{2}+\frac{2\ln n^2+\ln m}{\ln 2}\le\frac{5\kappa}{2}+8\ln m.$$

Finally, we bound the number of membership Oracle calls. At each iteration $i=1,\dots,n/4$, we update the weights of $|E_i|+|\mathcal{S}_i|=O\left(n^2\mathbf{p}+m\mathbf{q}\right)$ elements in expectation, each requiring one call to the membership Oracle. Thus in expectation, the total number of membership Oracle calls is $O\left(n\left(n^2\cdot\min\{n/\kappa^2\ln n,\,1\}+m\cdot\min\{n/\kappa^2\ln m,\,1\}\right)\right)$. ◀

Proof of Main Lemma. The proof is subdivided into three lemmas. For brevity, we set $t=n/4$. The first lemma is proved by examining the total weight of the sets of $\mathcal{S}$ in π_{t+1}.

▶ **Lemma 5.**

$$\mathbb{E}\left[\max_{S\in\mathcal{S}}\sum_{i=1}^{t}\mathrm{I}(e_i,S)\right]\le\frac{1}{\ln 2}\sum_{i=1}^{t}\mathbb{E}\left[\sum_{S\in\mathcal{S}}\frac{\pi_i(S)}{\pi_i(\mathcal{S})}\mathrm{I}(e_i,S)\right]+\frac{\kappa}{4}+\frac{\ln|\mathcal{S}|}{\ln 2}.$$

Proof. Let $\pi_{t+1}(\mathcal{S})$ denote the total weight of the sets of $\mathcal{S}$ in π_{t+1}. We bound $\pi_{t+1}(\mathcal{S})$ in two different ways. On the one hand, $\pi_{t+1}(\mathcal{S})$ is clearly lower-bounded by the weight of the set of maximum weight in π_{t+1}. Recall that the weight of a set S is doubled in iteration i if and only if S crosses e_i, therefore

$$\pi_{t+1}(\mathcal{S}) \geq \max_{S\in\mathcal{S}} \pi_{t+1}(S) = 2^{\max_{S\in\mathcal{S}} \sum_{i=1}^{t} \mathrm{I}(e_i,S)\cdot \mathbf{1}_{\{S\in\mathcal{S}_i\}}},$$

where $\mathbf{1}_{\mathcal{A}}$ denotes the indicator whether an event $\mathcal{A}$ happens. On the other hand, we can express $\pi_{t+1}(\mathcal{S})$ using the update rule of the algorithm

$$\begin{aligned}
\pi_{t+1}(\mathcal{S}) &= \sum_{S\in\mathcal{S}} \pi_{t+1}(S) = \sum_{S\in\mathcal{S}} \pi_t(S)\left(1+\mathrm{I}(e_t,S)\cdot\mathbf{1}_{\{S\in\mathcal{S}_t\}}\right)\\
&= \sum_{S\in\mathcal{S}} \pi_t(S) + \sum_{S\in\mathcal{S}} \pi_t(S)\mathrm{I}(e_t,S)\cdot\mathbf{1}_{\{S\in\mathcal{S}_t\}}\\
&= \pi_t(\mathcal{S}) + \pi_t(\mathcal{S})\sum_{S\in\mathcal{S}} \frac{\pi_t(S)}{\pi_t(\mathcal{S})}\mathrm{I}(e_t,S)\cdot\mathbf{1}_{\{S\in\mathcal{S}_t\}}\\
&= \pi_t(\mathcal{S})\left(1+\sum_{S\in\mathcal{S}} \frac{\pi_t(S)}{\pi_t(\mathcal{S})}\mathrm{I}(e_t,S)\cdot\mathbf{1}_{\{S\in\mathcal{S}_t\}}\right).
\end{aligned}$$

Unfolding this recursion and using the fact that $1+a\leq\exp(a)$, we get

$$\begin{aligned}
\pi_{t+1}(\mathcal{S}) &= \pi_1(\mathcal{S})\prod_{i=1}^{t}\left(1+\sum_{S\in\mathcal{S}} \frac{\pi_i(S)}{\pi_i(\mathcal{S})}\mathrm{I}(e_i,S)\cdot\mathbf{1}_{\{S\in\mathcal{S}_i\}}\right)\\
&\leq |\mathcal{S}|\cdot\exp\left(\sum_{i=1}^{t}\sum_{S\in\mathcal{S}} \frac{\pi_i(S)}{\pi_i(\mathcal{S})}\mathrm{I}(e_i,S)\cdot\mathbf{1}_{\{S\in\mathcal{S}_i\}}\right).
\end{aligned}$$

Putting together the obtained upper and lower bounds on $\pi_{t+1}(\mathcal{S})$, we get

$$2^{\max_{S\in\mathcal{S}} \sum_{i=1}^{t} \mathrm{I}(e_i,S)\cdot \mathbf{1}_{\{S\in\mathcal{S}_i\}}} \leq |\mathcal{S}|\cdot\exp\left(\sum_{i=1}^{t}\sum_{S\in\mathcal{S}} \frac{\pi_i(S)}{\pi_i(\mathcal{S})}\mathrm{I}(e_i,S)\cdot\mathbf{1}_{\{S\in\mathcal{S}_i\}}\right).$$

Taking the logarithm of each side yields

$$\ln(2)\cdot\max_{S\in\mathcal{S}}\sum_{i=1}^{t}\mathrm{I}(e_i,S)\cdot\mathbf{1}_{\{S\in\mathcal{S}_i\}} \leq \sum_{i=1}^{t}\sum_{S\in\mathcal{S}} \frac{\pi_i(S)}{\pi_i(\mathcal{S})}\mathrm{I}(e_i,S)\cdot\mathbf{1}_{\{S\in\mathcal{S}_i\}} + \ln|\mathcal{S}|. \tag{3}$$

If $\mathbf{q}=1$, then $\mathbf{1}_{\{S\in\mathcal{S}_i\}}=1$ for all i and $S\in\mathcal{S}$, thus taking total expectation we conclude

$$\mathbb{E}\left[\max_{S\in\mathcal{S}}\sum_{i=1}^{t}\mathrm{I}(e_i,S)\right] \leq \frac{1}{\ln 2}\sum_{i=1}^{t}\mathbb{E}\left[\sum_{S\in\mathcal{S}} \frac{\pi_i(S)}{\pi_i(\mathcal{S})}\mathrm{I}(e_i,S)\right] + \frac{\ln|\mathcal{S}|}{\ln 2}.$$

Assume that $\mathbf{q}<1$. Since $\max f(x)-\max g(x)\leq\max(f(x)-g(x))$, Equation (3) implies

$$\begin{aligned}
\ln(2)\cdot\mathbf{q}\cdot\max_{S\in\mathcal{S}}\sum_{i=1}^{t}\mathrm{I}(e_i,S) \leq{}& \ln(2)\cdot\max_{S\in\mathcal{S}}\sum_{i=1}^{t}\mathrm{I}(e_i,S)\cdot\left(\mathbf{q}-\mathbf{1}_{\{S\in\mathcal{S}_i\}}\right)\\
&+\sum_{i=1}^{t}\sum_{S\in\mathcal{S}} \frac{\pi_i(S)}{\pi_i(\mathcal{S})}\mathrm{I}(e_i,S)\cdot\mathbf{1}_{\{S\in\mathcal{S}_i\}} + \ln|\mathcal{S}|.
\end{aligned}$$

Taking total expectation of each side, we obtain

$$\mathbf{q}\ln(2)\cdot\mathbb{E}\left[\max_{S\in\mathcal{S}}\sum_{i=1}^{t}\mathrm{I}\left(e_i,S\right)\right]\leq\ln(2)\cdot\mathbb{E}\left[\max_{S\in\mathcal{S}}\sum_{i=1}^{t}\mathrm{I}\left(e_i,S\right)\cdot\left(\mathbf{q}-\mathbf{1}_{\{S\in\mathcal{S}_i\}}\right)\right]+\sum_{i=1}^{t}\sum_{S\in\mathcal{S}}\mathbb{E}\left[\frac{\pi_i(S)}{\pi_i(\mathcal{S})}\mathrm{I}\left(e_i,S\right)\cdot\mathbf{1}_{\{S\in\mathcal{S}_i\}}\right]+\ln|\mathcal{S}|\,. \tag{4}$$

Since for each fixed i, the random variables $\{\pi_i, e_i\}$ and $\mathcal{S}_i$ are independent, we get that

$$\sum_{i=1}^{t}\sum_{S\in\mathcal{S}}\mathbb{E}\left[\frac{\pi_i(S)}{\pi_i(\mathcal{S})}\mathrm{I}\left(e_i,S\right)\cdot\mathbf{1}_{\{S\in\mathcal{S}_i\}}\right]=\mathbf{q}\cdot\sum_{i=1}^{t}\sum_{S\in\mathcal{S}}\mathbb{E}\left[\frac{\pi_i(S)}{\pi_i(\mathcal{S})}\mathrm{I}\left(e_i,S\right)\right].$$

To bound the expectation of $\max_{S\in\mathcal{S}}\sum_{i=1}^{t}\mathrm{I}\left(e_i,S\right)\cdot\left(\mathbf{q}-\mathbf{1}_{\{S\in\mathcal{S}_i\}}\right)$, we will need the following concentration bound for martingales.

▶ **Lemma 6** (Freedman's inequality [15, 5]). *Let $Y_0,\ldots,Y_n$ be a martingale adapted to the filtration $F_0,\ldots,F_n$ such that $|Y_i-Y_{i-1}|<M$ for all i and $\sum_{i=1}^{n}\mathbb{E}\left[\,(Y_i-Y_{i-1})^2\mid F_{i-1}\,\right]\leq s$ almost surely for some $s>0$. Then for any $\varepsilon>0$,*

$$\mathbb{P}\left[\,Y_n-Y_0\geq\varepsilon\,\right]\leq\exp\left(-\frac{\varepsilon^2}{2(s+M\varepsilon)}\right).$$

▷ Claim 7.

$$\mathbb{P}\left[\max_{S\in\mathcal{S}}\sum_{i=1}^{t}\mathrm{I}\left(e_i,S\right)\cdot\left(\mathbf{q}-\mathbf{1}_{\{S\in\mathcal{S}_i\}}\right)\geq 2\sqrt{\mathbf{q}t\ln(|\mathcal{S}|t)}\right]\leq\frac{1}{t}.$$

Proof. For each $i\in[1,t]$ and $S\in\mathcal{S}$, consider the random variable $X_i(S)=\mathrm{I}\left(e_i,S\right)\cdot\left(\mathbf{q}-\mathbf{1}_{\{S\in\mathcal{S}_i\}}\right)$, which is measurable with respect to e_i and $\mathcal{S}_i$.
Let $F_i=\sigma\left(e_1,\ldots,e_i,S_1,\ldots,S_i,E_1,\ldots,E_i,\mathcal{S}_1,\ldots\mathcal{S}_i\right)$. Observe that conditioned on F_{i-1}, e_i and $\mathcal{S}_i$ are independent, and thus $\mathbb{E}\left[\,X_i(S)\mid F_{i-1}\,\right]=0$ for all $i\in[1,t]$ and $S\in\mathcal{S}$, as $\mathbb{E}\left[\,\mathbf{q}-\mathbf{1}_{\{S\in\mathcal{S}_i\}}\,\right]=0$. Therefore, $Y_0(S)=0$, $Y_k(S)=\sum_{i=1}^{k}X_i(S)$ is a martingale adapted to the filtration $F_0,\ldots,F_{k-1}$. Notice that for any $S\in\mathcal{S}$, $\sum_{i=1}^{t}\mathbb{E}[(Y_i(S)-Y_{i-1}(S))^2\mid F_{i-1}]\leq\mathbf{q}t$ and $|Y_i(S)-Y_{i-1}(S)|\leq 1$.
Thus Lemma 6 combined with the union bound implies for any $\varepsilon\leq\mathbf{q}t$,

$$\mathbb{P}\left(\max_{S\in\mathcal{S}}Y_t(S)\geq\varepsilon\right)\leq|\mathcal{S}|\exp\left(-\frac{\varepsilon^2}{4\mathbf{q}t}\right).$$

Setting $\varepsilon=2\sqrt{\mathbf{q}t\ln(|\mathcal{S}|t)}$, we conclude the proof of Claim 7. ◁

Applying Claim 7 and using that $\sum_{i=1}^{t}\mathrm{I}\left(e_i,S\right)\cdot\left(\mathbf{q}-\mathbf{1}_{\{S\in\mathcal{S}_i\}}\right)\leq t$ always holds, we get

$$\mathbb{E}\left[\max_{S\in\mathcal{S}}\sum_{i=1}^{t}\mathrm{I}\left(e_i,S\right)\cdot\left(\mathbf{q}-\mathbf{1}_{\{S\in\mathcal{S}_i\}}\right)\right]\leq 2\sqrt{\mathbf{q}t\ln(|\mathcal{S}|t)}+t\cdot\frac{1}{t}\leq 2\sqrt{\mathbf{q}t\ln(|\mathcal{S}|t)}+1.$$

Hence Equation (4) implies

$$\ln(2)\cdot\mathbf{q}\cdot\mathbb{E}\left[\max_{S\in\mathcal{S}}\sum_{i=1}^{t}\mathrm{I}\left(e_i,S\right)\right]$$

$$\leq \mathbf{q} \cdot \sum_{i=1}^{t} \mathbb{E}\left[\sum_{S \in \mathcal{S}} \frac{\pi_i(S)}{\pi_i(\mathcal{S})} \mathrm{I}\left(e_i, S\right) \right] + 2\sqrt{\mathbf{q} t \ln(|\mathcal{S}|t)} + 1 + \ln|\mathcal{S}|.$$

Dividing both sides by $\mathbf{q} \ln 2$ and substituting $\mathbf{q} = 39n\log(|\mathcal{S}|n/4)/\kappa^2 = 156t\log(|\mathcal{S}|t)/\kappa^2$,

$$\begin{aligned}
&\mathbb{E}\left[\max_{S \in \mathcal{S}} \sum_{i=1}^{t} \mathrm{I}\left(e_i, S\right) \right] \\
&\leq \frac{1}{\ln 2} \cdot \sum_{i=1}^{t} \mathbb{E}\left[\sum_{S \in \mathcal{S}} \frac{\pi_i(S)}{\pi_i(\mathcal{S})} \mathrm{I}\left(e_i, S\right) \right] + \frac{2}{\ln 2}\sqrt{\frac{t\ln(|\mathcal{S}|t)}{\mathbf{q}}} + \frac{1+\ln|\mathcal{S}|}{\mathbf{q}\ln 2} \\
&\leq \frac{1}{\ln 2} \cdot \sum_{i=1}^{t} \mathbb{E}\left[\sum_{S \in \mathcal{S}} \frac{\pi_i(S)}{\pi_i(\mathcal{S})} \mathrm{I}\left(e_i, S\right) \right] + \frac{2\kappa}{\sqrt{156}\ln 2} + \frac{1}{\ln 2} \cdot \frac{1+\ln|\mathcal{S}|}{\log(|\mathcal{S}|n/4)} \cdot \frac{\kappa^2}{78(n/2)} \\
&\leq \frac{1}{\ln 2} \cdot \sum_{i=1}^{t} \mathbb{E}\left[\sum_{S \in \mathcal{S}} \frac{\pi_i(S)}{\pi_i(\mathcal{S})} \mathrm{I}\left(e_i, S\right) \right] + \frac{\kappa}{4},
\end{aligned}$$

where we used that $\kappa \leq n/2$. This concludes the proof of Lemma 5. ◀

The next lemma is proven by applying analogous arguments for the total weight of edges in ω_{t+1} with a small adjustment as in each iteration we set some edge weights to zero. Recall that $\tilde{E}$ denotes the set of edges that have non-zero weight in ω_{t+1}.

▶ **Lemma 8.**

$$\sum_{i=1}^{t} \mathbb{E}\left[\sum_{e \in E} \frac{\omega_i(e)}{\omega_i(E)} \mathrm{I}\left(e, S_i\right) \right] < 2\ln(2) \cdot \mathbb{E}\left[\min_{e \in \tilde{E}} \sum_{i=1}^{t} \mathrm{I}\left(e, S_i\right) \right] + \frac{\kappa \ln 2}{4} + 2\ln|E|.$$

Proof. Let $\omega_{t+1}(E)$ denote the total weight of edges in ω_{t+1}. Again, we lower-bound $\omega_{t+1}(E)$ by the largest edge-weight in ω_{t+1}, which is now attained at some edge of $\tilde{E}$:

$$\omega_{t+1}(E) \geq \max_{e \in E} \omega_{t+1}(e) = \max_{e \in \tilde{E}} \omega_{t+1}(e) = \left(\frac{1}{2}\right)^{\min_{e \in \tilde{E}} \sum_{i=1}^{t} \mathrm{I}\left(e, S_i\right) \cdot \mathbf{1}_{\{e \in E_i\}}}.$$

The upper bound is obtained by using the algorithm's weight update rule. Since e_t has positive weight in ω_t, but its weight in ω_{t+1} is set to 0, we have a strict inequality

$$\begin{aligned}
\omega_{t+1}(E) = \sum_{e \in E} \omega_{t+1}(e) &< \sum_{e \in E} \omega_t(e) \left(1 - \frac{1}{2}\mathrm{I}\left(e, S_t\right) \cdot \mathbf{1}_{\{e \in E_t\}}\right) \\
&= \sum_{e \in E} \omega_t(e) - \frac{1}{2} \sum_{e \in E} \omega_t(e) \mathrm{I}\left(e, S_t\right) \cdot \mathbf{1}_{\{e \in E_t\}} \\
&= \omega_t(E) \left(1 - \frac{1}{2} \sum_{e \in E} \frac{\omega_t(e)}{\omega_t(E)} \mathrm{I}\left(e, S_t\right) \cdot \mathbf{1}_{\{e \in E_t\}}\right).
\end{aligned}$$

Unfolding this recursion and using the fact that $1 + a \leq \exp(a)$, we get

$$\omega_{t+1}(E) < |E| \cdot \exp\left(-\frac{1}{2} \sum_{i=1}^{t} \sum_{e \in E} \frac{\omega_i(e)}{\omega_i(E)} \mathrm{I}\left(e, S_i\right) \cdot \mathbf{1}_{\{e \in E_i\}}\right).$$

Combining the obtained upper and the lower bounds on $\omega_{t+1}(E)$ and taking the logarithm of each side, we get

$$\ln\left(\frac{1}{2}\right)\cdot\min_{e\in\tilde{E}}\sum_{i=1}^{t}\mathrm{I}\left(e,S_i\right)\cdot\mathbf{1}_{\{e\in E_i\}}<-\frac{1}{2}\sum_{i=1}^{t}\sum_{e\in E}\frac{\omega_i(e)}{\omega_i(E)}\mathrm{I}\left(e,S_i\right)\cdot\mathbf{1}_{\{e\in E_i\}}+\ln|E|,$$

which is equivalent to

$$\sum_{i=1}^{t}\sum_{e\in E}\frac{\omega_i(e)}{\omega_i(E)}\mathrm{I}\left(e,S_i\right)\cdot\mathbf{1}_{\{e\in E_i\}}<2\ln(2)\cdot\min_{e\in\tilde{E}}\sum_{i=1}^{t}\mathrm{I}\left(e,S_i\right)\cdot\mathbf{1}_{\{e\in E_i\}}+2\ln|E|. \tag{5}$$

If $\mathbf{p}=1$, then $\mathbf{1}_{\{e\in E_i\}}=1$ for all i and $e\in E$, thus taking total expectation we conclude

$$\sum_{i=1}^{t}\mathbb{E}\left[\sum_{e\in E}\frac{\omega_i(e)}{\omega_i(E)}\mathrm{I}\left(e,S_i\right)\right]<2\ln(2)\cdot\mathbb{E}\left[\min_{e\in\tilde{E}}\sum_{i=1}^{t}\mathrm{I}\left(e,S_i\right)\right]+2\ln|E|.$$

Assume that $\mathbf{p}<1$. Since $\min f(x)-\min g(x)\le\max(f(x)-g(x))$, Equation (5) implies

$$\begin{aligned}\sum_{i=1}^{t}\sum_{e\in E}\frac{\omega_i(e)}{\omega_i(E)}\mathrm{I}\left(e,S_i\right)\cdot\mathbf{1}_{\{e\in E_i\}}<\;&2\ln(2)\cdot\max_{e\in\tilde{E}}\sum_{i=1}^{t}\mathrm{I}\left(e,S_i\right)\cdot\left(\mathbf{1}_{\{e\in E_i\}}-\mathbf{p}\right)\\&+2\ln(2)\cdot\min_{e\in\tilde{E}}\sum_{i=1}^{t}\mathrm{I}\left(e,S_i\right)\cdot\mathbf{p}+2\ln|E|.\end{aligned}$$

Taking total expectation of each side, and using that for each fixed i, the random variables $\{\omega_i,S_i\}$ and E_i are independent, we get

$$\begin{aligned}\mathbf{p}\cdot\sum_{i=1}^{t}\sum_{e\in E}\mathbb{E}\left[\frac{\omega_i(e)}{\omega_i(E)}\mathrm{I}\left(e,S_i\right)\right]<\;&2\ln(2)\cdot\mathbb{E}\left[\max_{e\in\tilde{E}}\sum_{i=1}^{t}\mathrm{I}\left(e,S_i\right)\cdot\left(\mathbf{1}_{\{e\in E_i\}}-\mathbf{p}\right)\right]\\&+2\ln(2)\mathbf{p}\cdot\mathbb{E}\left[\min_{e\in\tilde{E}}\sum_{i=1}^{t}\mathrm{I}\left(e,S_i\right)\right]+2\ln|E|.\end{aligned} \tag{6}$$

We need the following claim whose proof uses Lemma 6 and is similar to Claim 7.

▷ Claim 9.

$$\mathbb{P}\left[\max_{e\in\tilde{E}}\sum_{i=1}^{t}\mathrm{I}\left(e,S_i\right)\cdot\left(\mathbf{1}_{\{e\in E_i\}-\mathbf{p}}\right)\ge2\sqrt{\mathbf{p}t\ln(|E|t)}\right]\le\frac{1}{t}.$$

This, together with the fact that $\sum_{i=1}^{t}\mathrm{I}\left(e,S_i\right)\cdot\left(\mathbf{1}_{\{e\in E_i\}}-\mathbf{p}\right)\le t$ always holds imply

$$\mathbb{E}\left[\max_{e\in\tilde{E}}\sum_{i=1}^{t}\mathrm{I}\left(e,S_i\right)\cdot\left(\mathbf{1}_{\{e\in E_i\}}\right)-\mathbf{p}\right]\le2\sqrt{\mathbf{p}t\ln(|E|t)}+t\cdot\frac{1}{t}\le2\sqrt{\mathbf{p}t\ln(|E|t)}+1.$$

Hence Equation (6) yields

$$\begin{aligned}&\mathbf{p}\cdot\sum_{i=1}^{t}\sum_{e\in E}\mathbb{E}\left[\frac{\omega_i(e)}{\omega_i(E)}\mathrm{I}\left(e,S_i\right)\right]\\&<2\ln(2)\mathbf{p}\cdot\mathbb{E}\left[\min_{e\in\tilde{E}}\sum_{i=1}^{t}\mathrm{I}\left(e,S_i\right)\right]+4\ln(2)\cdot\sqrt{\mathbf{p}t\ln(|E|t)}+2\ln2+2\ln|E|.\end{aligned}$$

Dividing both sides by $\mathbf{p} = 106n\log(|E|n/4)/\kappa^2 = 424t\log(|E|t)/\kappa^2$, we get

$$\begin{aligned}
&\sum_{i=1}^{t}\sum_{e\in E}\mathbb{E}\left[\frac{\omega_i(e)}{\omega_i(E)}\mathrm{I}\left(e,S_i\right)\right]\\
&< 2\ln(2)\cdot\mathbb{E}\left[\min_{e\in\tilde{E}}\sum_{i=1}^{t}\mathrm{I}\left(e,S_i\right)\right]+4\ln(2)\cdot\sqrt{\frac{t\ln(|E|t)}{\mathbf{p}}}+\frac{2\ln 2+2\ln|E|}{\mathbf{p}}\\
&= 2\ln(2)\cdot\mathbb{E}\left[\min_{e\in\tilde{E}}\sum_{i=1}^{t}\mathrm{I}\left(e,S_i\right)\right]+\frac{4\ln(2)\cdot\kappa}{\sqrt{424}}+\frac{2\ln 2+2\ln|E|}{2\log(|E|t)}\cdot\frac{\kappa^2}{n/2}\cdot\frac{1}{106}\\
&\leq 2\ln(2)\cdot\mathbb{E}\left[\min_{e\in\tilde{E}}\sum_{i=1}^{t}\mathrm{I}\left(e,S_i\right)\right]+\frac{\kappa\ln 2}{4},
\end{aligned}$$

where we used that $\kappa \leq n/2$. This concludes the proof of Lemma 8. ◀

We need one more lemma to tie the previous two together.

▶ **Lemma 10.** *For any $i \in [1,t]$, we have*

$$\mathbb{E}\left[\sum_{S\in\mathcal{S}}\frac{\pi_i(S)}{\pi_i(\mathcal{S})}\mathrm{I}\left(e_i,S\right)\right]=\mathbb{E}\left[\sum_{e\in E}\frac{\omega_i(e)}{\omega_i(E)}\mathrm{I}\left(e,S_i\right)\right].$$

Proof. Let $F_i = \sigma\left(e_1,\ldots,e_i,S_1,\ldots,S_i,E_1,\ldots,E_i,\mathcal{S}_1,\ldots\mathcal{S}_i\right)$. We have

$$\begin{aligned}
&\mathbb{E}\left[\sum_{S\in\mathcal{S}}\frac{\pi_i(S)}{\pi_i(\mathcal{S})}\mathrm{I}\left(e_i,S\right)\right]=\mathbb{E}\left[\mathbb{E}\left[\sum_{S\in\mathcal{S}}\frac{\pi_i(S)}{\pi_i(\mathcal{S})}\mathrm{I}\left(e_i,S\right)\,\middle|\,F_{i-1}\right]\right] \quad \text{and}\\
&\mathbb{E}\left[\sum_{e\in E}\frac{\omega_i(e)}{\omega_i(E)}\mathrm{I}\left(e,S_i\right)\right]=\mathbb{E}\left[\mathbb{E}\left[\sum_{e\in E}\frac{\omega_i(e)}{\omega_i(E)}\mathrm{I}\left(e,S_i\right)\,\middle|\,F_{i-1}\right]\right].
\end{aligned}$$

Observe that ω_i and π_i are measurable with respect to F_{i-1}, thus

$$\begin{aligned}
&\mathbb{E}\left[\sum_{S\in\mathcal{S}}\frac{\pi_i(S)}{\pi_i(\mathcal{S})}\mathrm{I}\left(e_i,S\right)\,\middle|\,F_{i-1}\right]=\sum_{e\in E}\frac{\omega_i(e)}{\omega_i(E)}\cdot\left(\sum_{S\in\mathcal{S}}\frac{\pi_i(S)}{\pi_i(\mathcal{S})}\mathrm{I}\left(e,S\right)\right)\\
&=\sum_{e\in E}\sum_{S\in\mathcal{S}}\frac{\omega_i(e)}{\omega_i(E)}\cdot\frac{\pi_i(S)}{\pi_i(\mathcal{S})}\mathrm{I}\left(e,S\right)\\
&=\sum_{S\in\mathcal{S}}\frac{\pi_i(S)}{\pi_i(\mathcal{S})}\cdot\left(\sum_{e\in E}\frac{\omega_i(e)}{\omega_i(E)}\mathrm{I}\left(e,S\right)\right)=\mathbb{E}\left[\sum_{e\in E}\frac{\omega_i(e)}{\omega_i(E)}\mathrm{I}\left(e,S_i\right)\,\middle|\,F_{i-1}\right].
\end{aligned}$$

◀

Finally, we combine Lemmas 5, 8, and 10 in the following way

$$\begin{aligned}
&\mathbb{E}\left[\max_{S\in\mathcal{S}}\sum_{i=1}^{t}\mathrm{I}\left(e_i,S\right)\right]\leq\frac{1}{\ln 2}\sum_{i=1}^{t}\mathbb{E}\left[\sum_{S\in\mathcal{S}}\frac{\pi_i(S)}{\pi_i(\mathcal{S})}\mathrm{I}\left(e_i,S\right)\right]+\frac{\kappa}{4}+\frac{\ln|\mathcal{S}|}{\ln 2} && \text{(Lemma 5)}\\
&=\frac{1}{\ln 2}\sum_{i=1}^{t}\mathbb{E}\left[\sum_{e\in E}\frac{\omega_i(e)}{\omega_i(E)}\mathrm{I}\left(e,S_i\right)\right]+\frac{\kappa}{4}+\frac{\ln|\mathcal{S}|}{\ln 2} && \text{(Lemma 10)}\\
&<\frac{1}{\ln 2}\left(2\ln(2)\cdot\mathbb{E}\left[\min_{e\in\tilde{E}}\sum_{i=1}^{t}\mathrm{I}\left(e,S_i\right)\right]+\frac{\kappa\ln 2}{4}+2\ln|E|\right)+\frac{\kappa}{4}+\frac{\ln|\mathcal{S}|}{\ln 2} && \text{(Lemma 8)}
\end{aligned}$$

$$\leq 2 \cdot \mathbb{E}\left[\min_{e\in\tilde{E}} \sum_{i=1}^{t} \mathrm{I}\left(e, S_i\right)\right] + \frac{\kappa}{2} + \frac{2\ln|E| + \ln|\mathcal{S}|}{\ln 2}.$$

This completes the proof of the Main Lemma and thus of Theorem 2. ◀

4 Corollaries of Theorem 1

Set systems with bounded dual shatter function. As before, let $(X, \mathcal{S})$ be a set system, $n = |X|$ and $m = |\mathcal{S}|$. We first recall the definition of the *dual shatter function* $\pi^*_{\mathcal{S}}$ of $(X, \mathcal{S})$. For any $\mathcal{R} \subseteq \mathcal{S}$, we say that the elements $x, y \in X$ are equivalent with respect to $\mathcal{R}$ if x belongs to the same sets of $\mathcal{R}$ as y. Then $\pi^*_{\mathcal{S}}(k)$ is defined as the maximum number of equivalence classes on X defined by a k-element subfamily $\mathcal{R} \subseteq \mathcal{S}$. The following theorem shows that set systems with polynomially bounded dual shatter function possess matchings with sublinear crossing number [23, Chap. 5.4].

▶ **Lemma 11.** *Let $(X, \mathcal{S})$ be a set system and c, d be constants such that $\pi^*_{\mathcal{S}}(k) \leq ck^d$ for all $k \in [1, n]$. Then there is a perfect matching of X such that any set $S \in \mathcal{S}$ crosses at most $c^{1/d} n^{1-1/d} + \ln m$ edges of the matching.*

Observe that by definition, the dual shatter function of $(Y, \mathcal{S}|_Y)$ is upper-bounded by the dual shatter function of $(X, \mathcal{S})$ for any $Y \subseteq X$. Thus Lemma 11 implies that any $Y \subseteq X$ has a perfect matching with crossing number at most $c^{1/d}|Y|^{1-1/d} + \ln m$ with respect to $\mathcal{S}$. Applying Theorem 1, we get the following corollary.

▶ **Corollary 12.** *Let $(X, \mathcal{S})$ be a set system and c, d be constants such that $\pi^*_{\mathcal{S}}(k) \leq ck^d$ for all $k \in [1, n]$. Then $\textsc{BuildMatching}\big((X, \mathcal{S}),\ c^{1/d},\ \ln m,\ 1 - \frac{1}{d}\big)$ returns a perfect matching of X with expected crossing number at most $\frac{5c^{1/d}d}{d-1} \cdot n^{1-1/d} + 11 \ln m \log n$ with an expected $\tilde{O}\left(mn^{2/d} + n^{2+2/d}\right)$ calls to the membership Oracle of $(X, \mathcal{S})$.*

Semialgebraic set systems. Let $\Gamma_{d,\Delta,s}$ denote the collection of semialgebraic sets in $\mathbb{R}^d$ that can be defined as the solution set of a Boolean combination of at most s polynomial inequalities of degree at most Δ. First, we give a bound on its dual shatter function.

▶ **Lemma 13.** *Let $(X, \mathcal{S})$ be a set system such that X is a set of points in $\mathbb{R}^d$ and each set in $\mathcal{S}$ is induced by an element $\Gamma_{d,\Delta,s}$. Then the dual shatter function of $(X, \mathcal{S})$ can be upper-bounded as $\pi^*_{\mathcal{S}}(k) \leq (4e\Delta s)^d \cdot k^d$.*

Proof. Let $\mathcal{R} \subseteq \Gamma_{d,\Delta,s}$ be a set of k ranges, defined by $\mathcal{P} = \{p_{ij} \ : \ 1 \leq i \leq k, 1 \leq j \leq s\}$, where each element is a d-variate polynomial of degree at most Δ. Observe that if $\operatorname{sign}[\, p(x)\,] = \operatorname{sign}[\, p(y)\,]$ for all $p \in \mathcal{P}$, then x, y are equivalent with respect to $\mathcal{R}$. Therefore, $\pi^*_{\Gamma_{d,\Delta,s}}(k)$ can be upper-bounded by the number of different sign patterns in $\{-1, 1\}^{ks}$ induced by ks d-variate polynomials of degree at most Δ. This quantity is bounded by $(4e\Delta s)^d \cdot k^d$, see [30, Theorem 3]. ◀

Now we can apply Corollary 12 and obtain the following.

▶ **Corollary 14.** *Let $(X, \mathcal{S})$ be a set system such that X is a set of n points in $\mathbb{R}^d$ and $\mathcal{S}$ consists of m subsets of X, each induced by an element of $\Gamma_{d,\Delta,s}$. Then $\textsc{BuildMatching}\big((X, \mathcal{S}),\ 4e\Delta s,\ \ln m, 1 - \frac{1}{d}\big)$ returns a perfect matching of X with expected crossing number at most $\frac{20e\Delta sd}{d-1} \cdot n^{1-1/d} + 11 \ln m \log n$ in expected time $\tilde{O}\left(s\Delta^d \left(mn^{2/d} + n^{2+2/d}\right)\right)$.*

Half-spaces. Let $\mathcal{H}_d$ denote the set of all half-spaces in $\mathbb{R}^d$ and consider set systems induced by $\mathcal{H}_d$. For this setting, a typical pre-processing step is constructing a small-sized subfamily of $\mathcal{H}_d$ – called a *test-set* – such that it suffices to construct a low-crossing matching with respect to this subfamily. We use a result of Matoušek [22] on test-sets, with a small addition:

▶ **Lemma 15** (Test set lemma [22]). *Let X be a set of n points in $\mathbb{R}^d$, $\mathcal{H}_d$ be the set of all half-spaces in $\mathbb{R}^d$, and t be a parameter. There exists a set $\mathcal{T}(t)$ of at most $(d+1)t^d$ hyperplanes such that if a perfect matching of X has crossing number κ with respect to $\mathcal{T}(t)$, then its crossing number with respect to $\mathcal{H}_d$ is at most $(d+1)\kappa + \frac{6d^2n}{t}$.*

Now let X be a set of n points in $\mathbb{R}^d$ and $\mathcal{T} = \mathcal{T}(n^{1/d})$ be the set of $(d+1)n$ half-spaces in $\mathbb{R}^d$ provided by Lemma 15. Notice that $\mathcal{T} \subset \mathcal{H}_d = \Gamma_{d,1,1}$, thus by Lemma 13, $\pi^*_{\mathcal{T}}(k) \leq (4e)^d k^d$. We apply Corollary 12 for $(X, \mathcal{T})$ and obtain the following.

▶ **Corollary 16.** *Let X be a set of n points in $\mathbb{R}^d$ and $\mathcal{T} = \mathcal{T}(n^{1/d})$ be the set of half-spaces provided by Lemma 15. Then* BuildMatching$\big((X,\mathcal{T}),\ 4e,\ \ln n, 1-\frac{1}{d}\big)$ *returns a perfect matching of X with expected crossing number at most $\left[\ 6d^2 + (d+1)\cdot\frac{20ed}{d-1}\ \right] n^{1-1/d} + \frac{11}{\ln 2}\ln^2 n$ with respect to half-spaces in $\mathbb{R}^d$, in expected time $O\left(dn^{2+2/d}\ln n\right)$.*

Balls. Let $\mathcal{B}_d$ denote the subsets of X that are induced by balls in $\mathbb{R}^d$. It is well known that there are mappings $\alpha : X \to \mathbb{R}^{d+1}$ and $\beta : \mathcal{B}_d \to \mathcal{H}_{d+1}$ such that for any $x \in X$ and $B \in \mathcal{B}_d$, we have $x \in B$ iff $\alpha(x) \in \beta(B)$, see eg. [24, Chap. 10]. This mapping and Lemma 15 applied in $\mathbb{R}^{d+1}$ with $t = n^{1/d}$ give the following test set lemma for $\mathcal{B}_d$.

▶ **Lemma 17.** *Let X be a set of n points in $\mathbb{R}^d$. There exists a set $\mathcal{Q}$ of at most $(d+2)n^{1+1/d}$ balls such that if a perfect matching of X has crossing number κ with respect to $\mathcal{Q}$, then its crossing number with respect to $\mathcal{B}_d$ is at most $(d+2)\kappa + 6(d+1)^2 n^{1-1/d}$.*

Given a set X of n points in $\mathbb{R}^d$, let $\mathcal{Q}$ be the set of balls provided by Lemma 15. As $\mathcal{Q} \subset \mathcal{B}_d \subset \Gamma_{d,2,1}$, the dual shatter function of $\mathcal{Q}$ can be bounded as $\pi^*_{\mathcal{Q}}(k) \leq (8e)^d k^d$ (Lemma 13). We apply Corollary 12 for $(X, \mathcal{Q})$, and obtain the following corollary.

▶ **Corollary 18.** *Let X be a set of n points in $\mathbb{R}^d$ and let $\mathcal{Q}$ be the set of balls provided by Lemma 17. Then* BuildMatching$\big((X,\mathcal{Q}),\ 8e,\ \ln(n^{1+1/d}),\ 1-\frac{1}{d}\big)$ *returns a perfect matching of X with expected crossing number at most $\left[\ 6(d+1)^2 + (d+2)\cdot\frac{40ed}{d-1}\ \right] n^{1-1/d} + \frac{11(d+1)}{d\ln 2}\ln^2 n$ with respect to balls in $\mathbb{R}^d$, in expected time $\tilde{O}(dn^{2+2/d})$.*

▶ Remark. The previous-best algorithm to construct spanning trees with crossing number $O(n^{1-1/d})$ with respect to $\mathcal{B}_d$ is based on randomized LP rounding and has time complexity $\tilde{O}(mn^2)$ [19, 11], which combined with Lemma 17 yields an $\tilde{O}\left(n^{3+1/d}\right)$ time algorithm. Alternatively, one can obtain a matching with suboptimal crossing number $O\left(n^{1-1/(d+1)}\right)$ by lifting X into $\mathbb{R}^{d+1}$, where the image of each range in $\mathcal{B}_d$ can be represented by a range in $\mathcal{H}_{d+1}$ and applying Chan's algorithm [8] with time complexity $\tilde{O}(n)$.

5 Empirical Aspects

In this section we present preliminary experimental results and provide some implementation details. We conducted our experiments on an accelerated version of BuildMatching (available on Github):

- instead of maintaining the weights on all the $O(n^2)$ edges, we work with an initial uniform random sample of $O(n \ln n)$ edges;
- at each iteration, we set $\mathbf{p} = \Theta\left(\frac{\ln n}{n^{1-1/d}}\right)$ and $\mathbf{q} = \Theta\left(\frac{\ln m}{n^{1-1/d}}\right)$ instead of $\mathbf{p} = \Theta\left(\frac{\ln n}{n^{1-2/d}}\right)$ and $\mathbf{q} = \Theta\left(\frac{\ln m}{n^{1-2/d}}\right)$.

Despite restricting ourselves to pick matching edges only from the inital sample of $O(n \log n)$ edges, we still obtain matchings with relatively low crossing numbers (see the table below). Incorporating this pre-sampling idea to the theoretical analysis of the algorithm is an interesting direction for future study.

Experimental setup. We apply the algorithm for set systems induced by half-spaces in dimensions $2, 4, 6, 8$, and 10. We consider two different types of input point sets:

`Grid`: each point is picked randomly in a cell of the uniform grid;

`Moment Curve`: each point is a slightly perturbed element of the moment curve.

All the experiments are performed with dual Xeon E5-2643 v3 processors, each with 6 cores, 12 threads, at 3.4 GHz.

`Grid`

Input size	$d=2$		$d=4$		$d=6$		$d=8$		$d=10$	
	cr #	time (s)	cr #	time (s)	cr #	time (s)	cr #	time (s)	cr #	time (s)
10000	162	58.89	699	11.84	1238	8.07	1639	6.38	1863	6.73
25000	330	279.82	1509	37.33	2804	26.49	3912	20.32	4525	20.76
50000	630	918.26	2732	99.62	5251	61.21	7387	47.02	8797	48.66
100000	1170	3001.16	5040	271.29	9774	147.91	13683	120.53	16754	110.48

`Moment Curve`

10000	57	58.51	324	11.68	807	7.9	1028	6.47	1354	6.12
25000	89	275.96	706	37.35	1698	24.08	2642	22.79	3411	20.62
50000	132	916.39	1151	98.25	2608	61.06	4836	52.4	6263	44.79
100000	209	2978.21	2797	268.95	5502	161.1	7743	133.25	10713	113.01

Evaluation. We present our experimental results in the table below. It shows the observed crossing numbers and running times on inputs of size up to 100000. We see that the algorithm becomes faster as the dimension increases (note that the crossing number increases with dimension). For example, in dimension 6, it takes only around 160 seconds to create a matching for 100000 points. Previous experimental results only considered inputs of size at most 159, see [16].

Test set generation. Linear-sized test set that achieves the guarantee of Lemma 15 can be constructed via cuttings, which are impractical in higher dimensions. Since the study of test-sets is not the main focus of this work and to speed-up the computations, our implementation, builds the test set by $n \log n$ random d-tuples of the input points; we report the crossing numbers with respect to this particular test set. We refer to [2] for a detailed overview on constructions and sizes of test-sets for various geometric objects.

6 Applications

We present an application of spanning path with low crossing number from learning theory. Further applications will be provided in the full version of the paper.

Approximating sign rank. Let $(X, \mathcal{S})$ be a set system and let $A \in \mathbb{R}^{n \times m}$ be its signed membership matrix, that is, $(A)_{x,S} = 1$ if $x \in S$ and $(A)_{x,S} = -1$ otherwise. The *sign rank* of $(X, \mathcal{S})$ is defined as the minimum rank of a matrix having the same sign pattern as A. Geometrically, it captures the minimum dimension of a Euclidean space in which $(X, \mathcal{S})$ can be embedded and realized by half-spaces through the origin. Using a connection between the sign-rank and the crossing number of a spanning path established in Alon et al.[7], we get the following corollary.

▶ **Corollary 19.** *Let $(X, \mathcal{S})$ be a set system and let $a > 0, b$ and $\gamma \in [1/\log n, 1]$ such that any $Y \subseteq X$ has a spanning path with crossing number at most $a|Y|^\gamma + b$. Then there is a randomized algorithm that constructs an embedding of X into $\mathbb{R}^D$ with $D \leq \frac{5}{\gamma} n^\gamma + (3b + 8 \ln m) \log n$ in expectation such that each $S \in \mathcal{S}$ can be represented with a half-space in $\mathbb{R}^D$. The algorithm makes $O\left(\min\left\{n^{4-2\gamma} \ln n + mn^{2-2\gamma} \ln m \ln n,\, n^3 + mn\right\}\right)$ calls to the membership Oracle of $(X, \mathcal{S})$.*

References

1. P. K. Agarwal. Simplex range searching. In *Journey Through Discrete Mathematics*, pages 1–30. Springer, 2017.
2. P. K. Agarwal and J. Matoušek. On range searching with semialgebraic sets. *Discrete & Computational Geometry*, 11(4):393–418, 1994.
3. P. K. Agarwal, J. Matoušek, and M. Sharir. On range searching with semialgebraic sets. II. *SIAM Journal on Computing*, 42(6):2039–2062, 2013.
4. P. K. Agarwal and J. Pan. Near-linear algorithms for geometric hitting sets and set covers. In *Proceedings of Symposium on Computational Geometry*, SOCG'14, page 271–279, 2014.
5. N. Alon, O. Ben-Eliezer, Y. Dagan, S. Moran, M. Naor, and E. Yogev. Adversarial laws of large numbers and optimal regret in online classification, 2021. `arXiv:2101.09054`.
6. N. Alon, D. Haussler, and E. Welzl. Partitioning and geometric embedding of range spaces of finite Vapnik-Chervonenkis dimension. In *SoCG '87*, 1987.
7. N. Alon, S. Moran, and A. Yehudayoff. Sign rank versus VC dimension. In *COLT*, 2016.
8. T. M. Chan. Optimal partition trees. *Discrete Comput. Geom.*, 47(4):661–690, 2012.
9. T. M. Chan, E. Grant, J. Könemann, and M. Sharpe. Weighted capacitated, priority, and geometric set cover via improved quasi-uniform sampling. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1576–1585, 2012.
10. B. Chazelle and E. Welzl. Quasi-optimal range searching in spaces of finite VC-dimension. *Discrete Comput. Geom.*, page 467–489, 1989.
11. C. Chekuri and K. Quanrud. Randomized MWU for positive LPs. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms*, SODA '18, page 358–377, 2018.
12. C. Chekuri, J. Vondrák, and R. Zenklusen. Dependent randomized rounding for matroid polytopes and applications. *arXiv preprint*, 2009. `arXiv:0909.4348`.
13. E. Ezra, S. Har-Peled, H. Kaplan, and M. Sharir. Decomposing arrangements of hyperplanes: VC-dimension, combinatorial dimension, and point location. *Discret. Comput. Geom.*, 64(1):109–173, 2020.
14. S. P. Fekete, M. E. Lübbecke, and H. Meijer. Minimizing the stabbing number of matchings, trees, and triangulations. In *Proceedings of Symposium on Discrete Algorithms (SODA)*, 2004.
15. D. A. Freedman. On Tail Probabilities for Martingales. *The Annals of Probability*, 3(1):100–118, 1975. `doi:10.1214/aop/1176996452`.
16. P. Giannopoulos, M. Konzack, and W Mulzer. Low-crossing spanning trees: an alternative proof and experiments. In *Proceedings of EuroCG*, 2014.
17. M. D. Grigoriadis and L. G. Khachiyan. A sublinear-time randomized approximation algorithm for matrix games. *Operations Research Letters*, 18(2):53–58, 1995.

18 S. Har-Peled. Constructing planar cuttings in theory and practice. *SIAM J. Comput.*, 29:2016–2039, 2000.

19 S. Har-Peled. Approximating spanning trees with low crossing number. *arXiv*, abs/0907.1131, 2009. `arXiv:0907.1131`.

20 D. Haussler. Sphere packing numbers for subsets of the boolean n-cube with bounded Vapnik-Chervonenkis dimension. *Journal of Combinatorial Theory, Series A*, 69(2):217–232, 1995.

21 M. Matheny and J. M. Phillips. Practical low-dimensional halfspace range space sampling. In *Annual European Symposium on Algorithms (ESA)*, volume 112, pages 62:1–62:14, 2018.

22 J. Matoušek. Efficient partition trees. *Discrete & Computational Geometry*, 8(3):315–334, 1992.

23 J. Matoušek. *Geometric Discrepancy: An Illustrated Guide*. Springer Berlin Heidelberg, 1999.

24 J. Matoušek. *Lectures on discrete geometry*, volume 212. Springer Science & Business Media, 2013.

25 J. Matoušek, E. Welzl, and L. Wernisch. Discrepancy and approximations for bounded VC-dimension. *Combinatorica*, 13(4):455–466, 1993.

26 N. H. Mustafa. Computing optimal epsilon-nets is as easy as finding an unhit set. In *46th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 87:1–87:12, 2019.

27 N. H. Mustafa, K. Dutta, and A. Ghosh. A simple proof of optimal epsilon-nets. *Combinatorica*, 38(5):1269–1277, 2018.

28 N. H. Mustafa and K. Varadarajan. Epsilon-approximations and Epsilon-nets. In J. E. Goodman, J. O'Rourke, and C. D. Tóth, editors, *Handbook of Discrete and Computational Geometry*. CRC Press LLC, 2017.

29 K. R. Varadarajan. Weighted geometric set cover via quasi-uniform sampling. In *Proceedings of ACM Symposium on Theory of Computing (STOC)*, pages 641–648, 2010.

30 H. E. Warren. Lower bounds for approximation by nonlinear manifolds. *Transactions of the American Mathematical Society*, 133(1):167–178, 1968.

31 E. Welzl. Partition trees for triangle counting and other range searching problems. In *Proceedings of Annual Symposium on Computational Geometry (SoCG)*, page 23–33, 1988.

32 E. Welzl. On spanning trees with low crossing numbers. In *Data Structures and Efficient Algorithms, Final Report on the DFG Special Joint Initiative*, page 233–249, 1992.

Colouring Polygon Visibility Graphs and Their Generalizations

James Davies
Department of Combinatorics and Optimization, School of Mathematics,
University of Waterloo, Canada

Tomasz Krawczyk
Department of Theoretical Computer Science, Faculty of Mathematics and Computer Science,
Jagiellonian University, Kraków, Poland

Rose McCarty
Department of Combinatorics and Optimization, School of Mathematics,
University of Waterloo, Canada

Bartosz Walczak
Department of Theoretical Computer Science, Faculty of Mathematics and Computer Science,
Jagiellonian University, Kraków, Poland

Abstract

Curve pseudo-visibility graphs generalize polygon and pseudo-polygon visibility graphs and form a hereditary class of graphs. We prove that every curve pseudo-visibility graph with clique number ω has chromatic number at most $3 \cdot 4^{\omega-1}$. The proof is carried through in the setting of ordered graphs; we identify two conditions satisfied by every curve pseudo-visibility graph (considered as an ordered graph) and prove that they are sufficient for the claimed bound. The proof is algorithmic: both the clique number and a colouring with the claimed number of colours can be computed in polynomial time.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Visibility graphs, χ-boundedness, pseudoline arrangements, ordered graphs

Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.29

Related Version *Full Version*: `https://arxiv.org/abs/2103.07803`

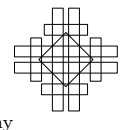
Funding Tomasz Krawczyk and Bartosz Walczak were partially supported by the National Science Centre of Poland grant 2015/17/D/ST1/00585.

Acknowledgements We thank Bodhayan Roy for sharing the problem of whether polygon visibility graphs are χ-bounded.

1 Introduction

A *polygon* is a Jordan curve made of finitely many line segments. A *polygon visibility graph* is the graph on the set of vertices of a polygon P that has an edge between each pair of *mutually visible* vertices, which means that the line segment connecting them is disjoint from the exterior of P. A class of graphs is *χ-bounded* if there is a function that bounds the chromatic number in terms of the clique number for every graph in the class. A clique in a polygon visibility graph has a natural interpretation – it is the maximum size of a subset of the vertices whose convex hull is disjoint from the exterior of the polygon (see Figure 1, top-left). The starting point of and main motivation for this work is the question of Kára, Pór, and Wood [23] of whether the class of polygon visibility graphs is χ-bounded. We answer it in the affirmative.

▶ **Theorem 1.1.** *Every polygon visibility graph with clique number ω has chromatic number at most $3 \cdot 4^{\omega-1}$.*



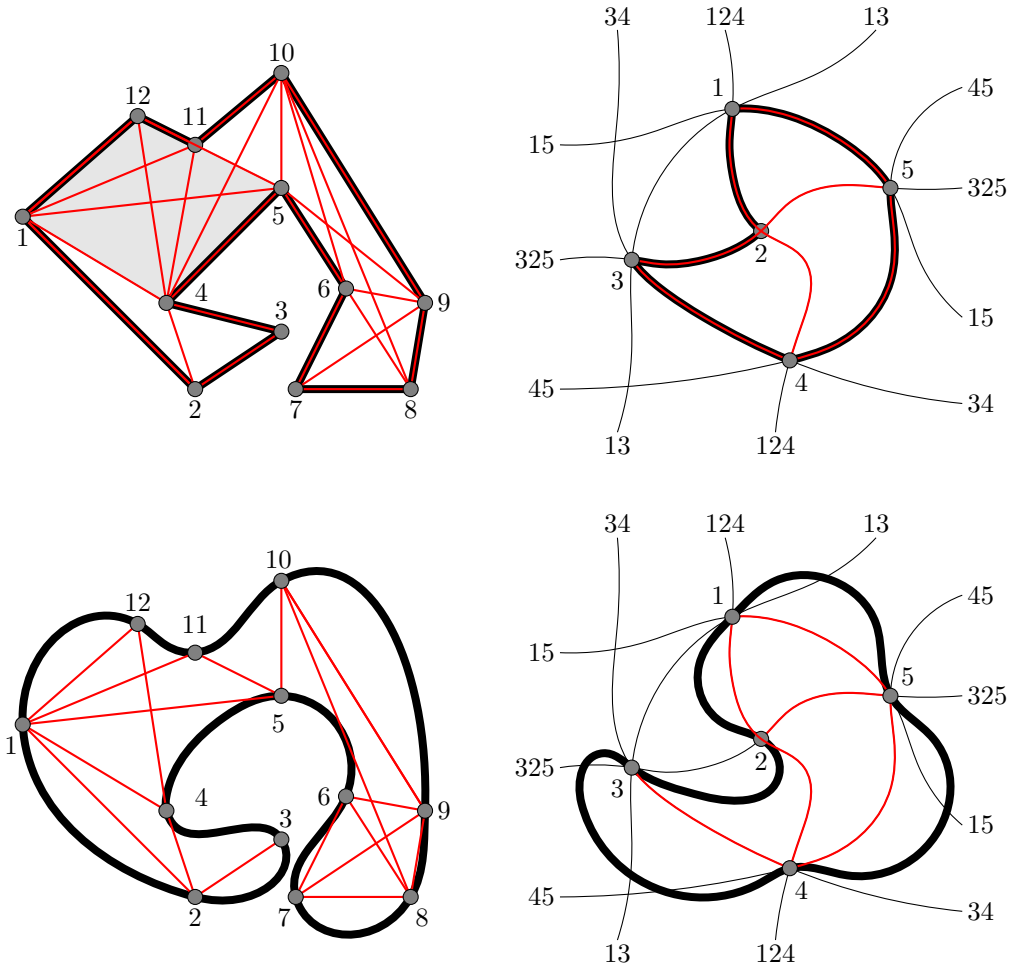


Figure 1 From left to right: a polygon visibility graph (where the convex hull of a maximum clique is shaded), a pseudo-polygon visibility graph, a curve visibility graph, and a curve pseudo-visibility graph. A "visibility" between each pair of adjacent vertices is drawn with a red (pseudo-)segment.

The bound in Theorem 1.1 also holds for all induced subgraphs of polygon visibility graphs. Such graphs can be defined alternatively as *curve visibility graphs*, that is, visibility graphs of points on a Jordan curve, where two points are considered to be mutually visible if the line segment connecting them is disjoint from the exterior of the curve (see Figure 1, bottom-left).

O'Rourke and Streinu [26] studied visibility graphs of pseudo-polygons (polygons on pseudoline arrangements; see Figure 1, top-right), where two vertices of the polygon are considered to be mutually visible if the pseudoline segment connecting them in the arrangement is disjoint from the exterior of the polygon. As a common generalization of these graphs and curve visibility graphs, we define curve pseudo-visibility graphs as follows. For a pseudoline arrangement $\mathcal{L}$, a Jordan curve K, and a finite set V of points on K any two of which lie on a common pseudoline in $\mathcal{L}$, the *curve pseudo-visibility graph* $G_{\mathcal{L}}(K, V)$ has vertex set V and has an edge between each pair of vertices such that the pseudoline segment in $\mathcal{L}$ connecting them is disjoint from the exterior of K (see Figure 1, bottom-right). We elaborate on this notion in Section 2; in particular, we show that curve pseudo-visibility graphs are exactly the induced subgraphs of the visibility graphs of pseudo-polygons. With this notion in hand, we provide the following topological generalization of Theorem 1.1.

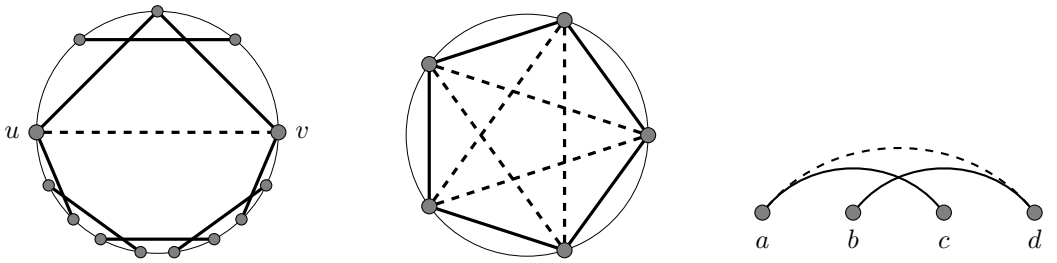


Figure 2 A graph in $\mathcal{H}$ (left), an ordered hole (middle), and the forbidden configuration for a capped graph (right). Dashed lines indicate non-edges. The pairs of vertices where no lines are drawn can be edges or non-edges.

▶ **Theorem 1.2.** *Every curve pseudo-visibility graph with clique number ω has chromatic number at most $3 \cdot 4^{\omega-1}$.*

To prove Theorem 1.2 (and thus Theorem 1.1), we turn our attention to ordered graphs, where an *ordered graph* is a pair $(G, \prec)$ such that G is a graph and $\prec$ is a linear order on the vertices of G. A curve pseudo-visibility graph comes with a natural linear order on the vertices (determined up to rotation), which makes it an ordered graph; it is the order in which the vertices are encountered when following the Jordan curve in the counterclockwise direction starting from an arbitrarily chosen vertex. An ordered graph $(H, \prec_H)$ is an (*induced*) *ordered subgraph* of an ordered graph $(G, \prec)$ if H is a subgraph (an induced subgraph, respectively) of G and $\prec_H$ is the restriction of $\prec$ to the vertices of H. There are two natural families of ordered obstructions to (that is, ordered graphs that cannot occur as induced ordered subgraphs of) curve pseudo-visibility graphs: the family $\mathcal{H}$ that we define in Section 3 and the family of ordered holes (see Figure 2), both easily verifiable in polynomial time. We prove the following further generalization of Theorem 1.2.

▶ **Theorem 1.3.** *Every $\mathcal{H}$-free ordered graph with clique number $\omega \geqslant 2$ has chromatic number at most $3 \cdot 4^{\omega}(\omega - 1)$ in general and at most $3 \cdot 4^{\omega-1}$ when also ordered-hole-free. Moreover, there is a polynomial-time algorithm that takes in an $\mathcal{H}$-free ordered graph and computes its clique number ω and a colouring with the claimed number of colours.*

Our proofs of Theorems 1.1–1.3 ultimately lead to the class of capped graphs, which may be of independent interest. A *capped graph* is an ordered graph $(G, \prec)$ such that for any four vertices $a \prec b \prec c \prec d$, if $ac, bd \in E(G)$, then $ad \in E(G)$; see Figure 2 (right). (This condition was previously studied for terrain visibility graphs [1, 3] as the "X-property".) We show that the vertices of any $\mathcal{H}$-free ordered graph can be partitioned into three sets each inducing a capped graph. This way, Theorem 1.3 becomes a corollary to the following.

▶ **Theorem 1.4.** *Every capped graph with clique number $\omega \geqslant 2$ has chromatic number at most $4^{\omega}(\omega - 1)$ in general and at most $4^{\omega-1}$ when also ordered-hole-free. Moreover, there is a polynomial-time algorithm that takes in a capped graph and computes its clique number ω and a colouring with the claimed number of colours.*

Any improvement on the bounds in Theorem 1.4 would immediately imply corresponding improvements in Theorems 1.1–1.3. A major open problem for most known χ-bounded classes of graphs is whether they are polynomially χ-bounded, that is, whether the chromatic number of the graphs in the class is bounded by a polynomial function of their clique number. Esperet [17] conjectured that *every* hereditary class of graphs that is χ-bounded is polynomially χ-bounded. While we have little faith in this conjecture, we do expect that it holds for capped graphs (and, consequently, for the graphs considered in Theorems 1.1–1.3).

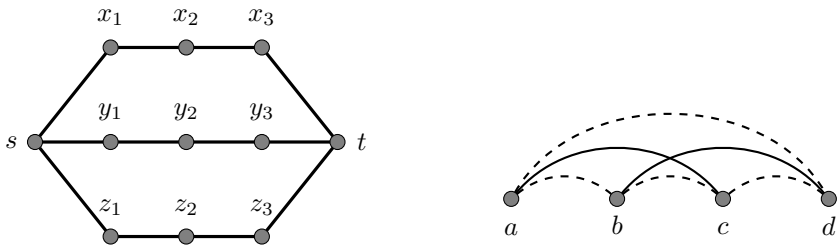


Figure 3 The banana B_4 (left) and the ordered graph X (right).

▶ **Conjecture 1.5.** *There is a polynomial function p such that every capped graph with clique number ω has chromatic number at most $p(\omega)$.*

While our proof of Theorem 1.4 is direct, we remark that a recent result of Scott and Seymour [32] implies χ-boundedness (with a much weaker bound) of the significantly broader class of X-free ordered graphs, that is, ordered graphs excluding the four-vertex ordered graph X illustrated in Figure 3 (right) as an induced ordered subgraph. In particular, every capped graph is X-free. Tomon [35] conjectured that the class of X-free ordered graphs is χ-bounded. This statement implies not only Theorem 1.4 but also the theorem of Rok and Walczak [31] that so-called outerstring graphs are χ-bounded. This is because outerstring graphs (with the natural linear order on the vertices) are easily seen to be X-free. Scott and Seymour [32] proved that for every graph H that is a "banana" (or more generally – a "banana tree"), the class of graphs excluding all subdivisions of H as induced subgraphs is χ-bounded. Figure 3 (left) shows an example of a "banana" B_4 with the property that no subdivision of B_4 can be made X-free under any order of the vertices. This shows that the aforementioned result of Scott and Seymour implies Tomon's conjecture.

▶ **Theorem 1.6.** *The class of X-free ordered graphs is χ-bounded.*

We present a detailed proof of Theorem 1.6 in the full version. We conclude the introduction with a brief literature review in order to place Theorems 1.1–1.4 and 1.6 in context.

Geometric graph classes and χ-boundedness

Various classic examples of χ-bounded graph classes are defined in terms of geometric representations. For instance, intersection graphs of axis-parallel rectangles [5] and circle graphs [22] are χ-bounded. Most of the literature in this direction focuses on intersection or disjointness graphs of objects in the plane. While the class of intersection graphs of curves in the plane is not itself χ-bounded [28], some very general subclasses are [14, 30]. There are also very precise results for disjointness graphs of certain kinds of curves in the plane [27].

Less is known about χ-boundedness of visibility graphs, even though various kinds of such graphs have been considered in the literature – see [20] for a survey. Kára, Pór, and Wood [23] conjectured that the class of point visibility graphs is χ-bounded, but this was disproven by Pfender [29]. Some types of bar visibility graphs are related to interval graphs [15] and planar graphs [25] and are therefore known to be χ-bounded.

Axenovich, Rollin, and Ueckerdt [7] considered the problem of whether ordered graphs excluding a fixed ordered graph $(H, \prec)$ as an ordered subgraph (not necessarily induced) have bounded chromatic number; they showed various cases of $(H, \prec)$ for which the answers are positive and negative. In particular, the answer is negative if H contains a cycle (as it is for unordered graphs), but they showed it is also negative for some acyclic ordered graphs

$(H, \prec)$. Pach and Tomon [27] used some specific classes of forbidden induced ordered graphs as a tool for studying χ-boundedness of disjointness graphs of curves. Max point-tolerance graphs [13] and classes of graphs of bounded twin-width [9] are also known to be χ-bounded and have well-understood characterizations as ordered graphs.

Algorithmic considerations

The class of curve pseudo-visibility graphs is hereditary, whereas most well-known classes of visibility graphs are not, including the classes of point visibility graphs, polygon visibility graphs, and pseudo-visibility graphs. The condition of the class being hereditary is very natural to impose when studying χ-boundedness and implies that curve pseudo-visibility graphs can be characterized by excluded induced (ordered) subgraphs. There has been a good deal of work on the characterization and recognition problems, but for point visibility graphs and polygon visibility graphs the problems appear to be hard (see [12] and [19]).

These difficult characterization problems tend to become tractable, and have more natural solutions, in the "pseudo-visibility setting" [1, 2, 18, 26]. This is due to the connection between stretchability of pseudoline arrangements and representability of rank-3 oriented matroids. So the pseudo-visibility setting is more combinatorial because it suffices to find the associated rank-3 oriented matroid without worrying about representability. Representability provides a real difficulty; the pseudo-visibility setting is strictly more general for polygon visibility graphs [33], even when certain restrictions are imposed [3].

It is an interesting problem to characterize ordered curve pseudo-visibility graphs by excluded induced ordered subgraphs. The two aforementioned classes of obstructions ($\mathcal{H}$ and the ordered holes) are likely to be insufficient for a full characterization – they roughly correspond to the first two of the four necessary conditions for an ordered graph to be a polygon visibility graph described by Ghosh [19]. Nevertheless, we conjecture the following.

▶ **Conjecture 1.7.** *Ordered curve pseudo-visibility graphs can be recognized in polynomial time.*

The part of Theorem 1.3 concerning polynomial-time computation of clique number extends well-known results regarding polygon visibility graphs [6, 16, 21], although our algorithm is certainly slower. We cannot expect to get an exact algorithm for the chromatic number, as Çağırıcı, Hliněný, and Roy [11] proved that it is NP-complete to decide if a polygon visibility graph is 5-colourable, even when the polygon is provided as part of the input.

2 Curve pseudo-visibility graphs

A *pseudoline* is a simple curve which separates the plane into two unbounded regions. A *pseudoline arrangement* is a set of pseudolines such that each pair intersects in exactly one point, where they cross. A *pseudo-configuration* is a pair $(\mathcal{L}, V)$ such that $\mathcal{L}$ is a pseudoline arrangement and V is a (finite) set of points on $\bigcup \mathcal{L}$ with the property that any two points in V lie on a common pseudoline in $\mathcal{L}$ (which is therefore unique). A pseudo-configuration $(\mathcal{L}, V)$ is in *general position* if no three points in V lie on a common pseudoline in $\mathcal{L}$.

Let $(\mathcal{L}, V)$ be a pseudo-configuration and K be a Jordan curve passing through all points in V. The *exterior* of K is the unbounded component of $\mathbb{R}^2 \setminus K$. We say that two points $u, v \in V$ are *mutually visible* in K if the pseudoline segment in $\mathcal{L}$ connecting u and v is disjoint from the exterior of K. The *curve pseudo-visibility graph* $G_{\mathcal{L}}(K, V)$ has vertex set V and has an edge uv for each pair of vertices $u, v \in V$ that are mutually visible in K. The curve K is a *pseudo-polygon* on $\mathcal{L}$ with vertex set V if every segment of K between two consecutive points in V is contained in a single pseudoline in $\mathcal{L}$. Graphs of the form

$G_{\mathcal{L}}(K, V)$ where K is a pseudo-polygon on $\mathcal{L}$ with vertex set V and $(\mathcal{L}, V)$ is in general position were considered by O'Rourke and Streinu [26] as *pseudo-polygon visibility graphs*. As we will see, the general position assumption is not actually restrictive in this setting.

The following two propositions imply that curve pseudo-visibility graphs are exactly the induced subgraphs of pseudo-polygon visibility graphs. First we find a pseudo-polygon, and then we take care of the general position assumption.

▶ **Proposition 2.1.** *For every curve pseudo-visibility graph $G = G_{\mathcal{L}}(K, V)$, there exist a pseudo-configuration $(\mathcal{L}', V')$ and a pseudo-polygon K' on $\mathcal{L}'$ with vertex set V' such that $\mathcal{L} \subseteq \mathcal{L}'$, $V \subseteq V'$, the points in V occur in the same cyclic order on K' as on K, and G is the subgraph of $G_{\mathcal{L}'}(K', V')$ induced on V.*

Proof. We can assume that K intersects $\bigcup \mathcal{L}$ only finitely many times. To see this, consider the finite plane graph H with a vertex for each intersection point of two pseudolines in $\mathcal{L}$ (including the points in V) and with an edge for each pseudoline segment in $\mathcal{L}$ connecting two vertices and passing through no other vertex. Let H' be the vertex-spanning subgraph of H obtained by including only the edges whose pseudoline segment is disjoint from the exterior of K. Thus K is contained in the closure of the outer (unbounded) face of H'. By following the boundary of this outer face very closely (and making thin connections between connected components of the boundary if H' is disconnected), we can choose K to intersect $\bigcup \mathcal{L}$ only finitely many times while preserving the graph $G_{\mathcal{L}}(K, V)$ and the order of points on K.

Let $(\mathcal{L}^*, V^*)$ be a pseudo-configuration such that $\mathcal{L} \subset \mathcal{L}^*$, $V \subset V^* \subset K$, every open segment of K connecting two points in $\bigcup \mathcal{L}$ contains a point in $V^* \setminus \bigcup \mathcal{L}$, each point in V^* lies on at least two pseudolines in $\mathcal{L}^*$, and $|V^*| \geqslant 3$; we first select V^* and then extend $\mathcal{L}$ to $\mathcal{L}^*$ using Levi's extension lemma [24]. As before, we can assume that K intersects $\bigcup \mathcal{L}^*$ only finitely many times. We further assume that K has the minimum number of intersection points with $\bigcup \mathcal{L}^*$ among all Jordan curves K^* that pass through all of the points in V^* in the same order as K and are such that $G = G_{\mathcal{L}}(K^*, V)$.

Let $V' = K \cap \bigcup \mathcal{L}^*$. In particular, $V^* \subseteq V'$. We extend $\mathcal{L}^*$ to a family of pseudolines $\mathcal{L}'$ such that $(\mathcal{L}', V')$ is a pseudo-configuration, using Levi's extension lemma [24]. For any two points $u, v \in V'$ consecutive on K, let K_{uv} be the segment of K between u and v (which is internally disjoint from $\bigcup \mathcal{L}^*$), let L'_{uv} be the pseudoline in $\mathcal{L}'$ passing through u and v, let K'_{uv} be the segment uv of L'_{uv}, and let E_{uv} be the unbounded component of $\mathbb{R}^2 \setminus (K_{uv} \cup K'_{uv})$. To construct K', we replace K_{uv} by K'_{uv} for every pair of points $u, v \in V'$ consecutive on K. Since any pseudoline in $\mathcal{L}^*$ intersecting K'_{uv} needs to intersect K_{uv}, every pseudoline in $\mathcal{L}^* \setminus \{L'_{uv}\}$ is fully contained in $E_{uv} \cup \{u, v\}$. Consequently, since each point in V^* lies on at least two pseudolines in $\mathcal{L}^*$, we have $V^* \subset E_{uv} \cup \{u, v\}$.

We claim that $V' \subset E_{uv} \cup \{u, v\}$ as well. If $L'_{uv} \notin \mathcal{L}^*$, then indeed $V' \subset \bigcup \mathcal{L}^* \subset E_{uv} \cup \{u, v\}$. Now, suppose $L'_{uv} \in \mathcal{L}^*$. We have $u \notin \bigcup \mathcal{L}$ or $v \notin \bigcup \mathcal{L}$ by the choice of V^*, and thus $L'_{uv} \notin \mathcal{L}$. Suppose $K \setminus K_{uv} \not\subset E_{uv}$. Since $\bigcup \mathcal{L} \subseteq \bigcup(\mathcal{L}^* \setminus \{L'_{uv}\}) \subset E_{uv} \cup \{u, v\}$, $V^* \subset E_{uv} \cup \{u, v\}$, and $K \setminus K_{uv}$ is disjoint from K_{uv}, the parts of $K \setminus K_{uv}$ not lying in E_{uv} can be moved into E_{uv} decreasing the number of intersection points with $\bigcup \mathcal{L}^*$ (as $|V^*| \geqslant 3$) while preserving the graph $G_{\mathcal{L}}(K, V)$, which contradicts the choice of K. Thus $V' \subset (K \setminus K_{uv}) \cup \{u, v\} \subset E_{uv} \cup \{u, v\}$ when $L'_{uv} \in \mathcal{L}^*$.

For any two pairs $u, v \in V'$ and $u', v' \in V'$ of consecutive points on K, if the internal parts of K'_{uv} and $K'_{u'v'}$ intersect, then the four points u, u', v, v' occur in this or the reverse order on the boundary of $(E_{uv} \cup \{u, v\}) \cap (E_{u'v'} \cup \{u', v'\})$, so the internal parts of K_{uv} and $K_{u'v'}$ intersect, which is impossible. Thus K' is a Jordan curve – a pseudo-polygon on $\mathcal{L}'$ with vertex set V'. Furthermore, $\bigcup \mathcal{L} \subset \bigcap_{uv}(E_{uv} \cup \{u, v\})$, which implies $G_{\mathcal{L}}(K', V) = G_{\mathcal{L}}(K, V)$. ◀

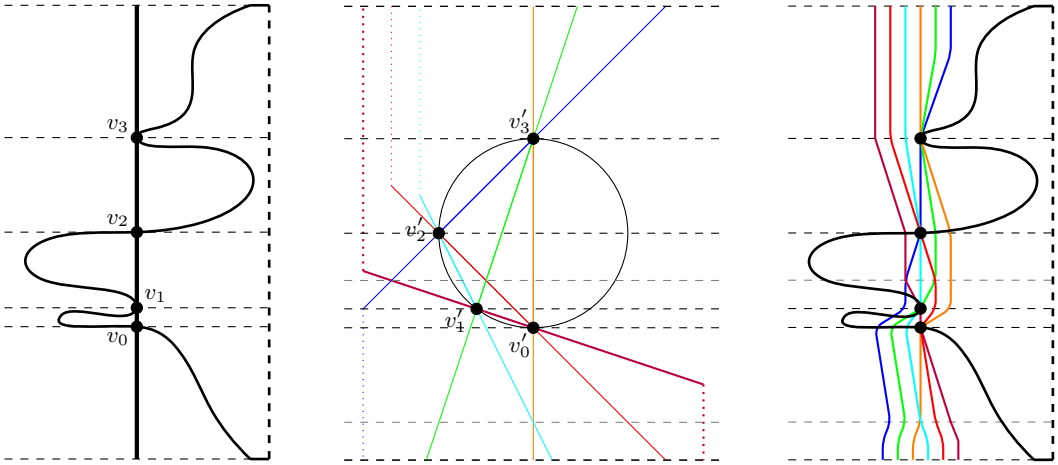


Figure 4 Replacing L with a bundle of pseudo-lines $\mathcal{B}_L$ in the proof of Proposition 2.2.

▶ **Proposition 2.2.** *For every curve pseudo-visibility graph $G = G_{\mathcal{L}}(K, V)$, there exist a pseudo-configuration $(\mathcal{L}', V')$ in general position and a pseudo-polygon K' on $\mathcal{L}'$ with vertex set V' such that $V \subseteq V'$, the points in V occur in the same cyclic order on K' as on K, and G is the subgraph of $G_{\mathcal{L}'}(K', V')$ induced on V.*

Proof. By Proposition 2.1, we can assume without loss of generality that K is a pseudo-polygon on $\mathcal{L}$. Suppose there is a pseudoline L in $\mathcal{L}$ passing through more than two points in V. We show that L can be replaced in $\mathcal{L}$ by a bunch $\mathcal{B}_L$ of pseudolines in a small neighbourhood of L so that the set $(\mathcal{L} \setminus \{L\}) \cup \mathcal{B}_L$ is a pseudoline arrangement and the following conditions hold for any two distinct points $u, v \in V \cap L$.

1. There is a pseudoline $L_{uv} \in \mathcal{B}_L$ passing through u, v, and no other points in V.
2. If u and v are consecutive points of $V \cap L$ on L, then the segment uv of L_{uv} coincides with the segment uv of L.
3. If the segment uv of L is disjoint from the exterior of K, then so is the segment uv of L_{uv}.
4. If the segment uv of L intersects the exterior of K, then so does the segment uv of L_{uv}.

Condition 4 is automatically satisfied whenever we make $\mathcal{B}_L$ lie in a sufficiently small neighbourhood of L. Applying this replacement repeatedly for every such pseudoline L yields a claimed pseudoline arrangement $\mathcal{L}'$.

For the replacement step, assume without loss of generality that L is a vertical line (by applying an appropriate homeomorphism of the plane before and the inverse homeomorphism after the step). Enumerate the points in $V \cap L$ as $v_0, \ldots, v_k$ from bottom to top. Let C be the circle with vertical diameter v_0v_k. Let $v'_0 = v_0$ and $v'_k = v_k$. For $0 < i < k$, let H_i be the horizontal line through v_i, and let v'_i be the left/the right/any intersection point of C and H_i if the exterior of K touches v_i from the left side/the right side/both sides of the vertical line L (respectively). For $0 \leqslant i < j \leqslant k$, let $L'_{i,j}$ be the straight line passing through v'_i and v'_j. The bundle $\mathcal{B}_L$ is obtained by "flattening" the family of lines $\{L_{i,j}\}_{0 \leqslant i < j \leqslant k}$ horizontally to fit it in a small neighbourhood of L and performing local horizontal shifts to guarantee conditions 1 and 2; condition 3 then follows. See Figure 4 for an illustration. ◀

Recall that an *ordered graph* is a tuple $(G, \prec)$ such that G is a graph and $\prec$ is a linear order on its vertex set. While it is more convenient to work with linear orders, the points on a Jordan curve are really ordered cyclically. A *rotation* of a linear order $\prec$ is any linear order obtained from $\prec$ by repeatedly making the largest element the smallest. We think of

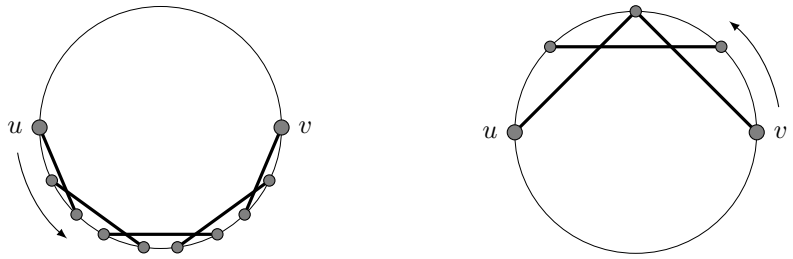


Figure 5 A crossing sequence from u to v (left) and from v to u (right).

any finite set of points V on a Jordan curve K as being ordered counterclockwise around K, as in Figure 1 (bottom-left). We call any linear order which begins at an arbitrary point in V and then follows K in the counterclockwise direction a *natural order* of V on K. A curve pseudo-visibility graph $G_{\mathcal{L}}(K, V)$ along with a natural order of V on K forms an *ordered curve pseudo-visibility graph*.

If $(G, \prec)$ is an ordered graph with vertices $a \prec b \prec c \prec d$ and edges ac and bd, we say that ac *crosses* bd, that ac and bd are *crossing edges*, and that bd is *crossed* by ac. Two edges which are not crossing are called *non-crossing*. The property that a pair of edges is crossing/non-crossing is preserved under rotation (since, using this terminology, we do not specify which edge crosses the other). In particular, it is well defined for an ordered curve pseudo-visibility graph regardless of the choice of a natural ordering.

▶ **Lemma 2.3.** *For a curve pseudo-visibility graph $G_{\mathcal{L}}(K, V)$ with $(\mathcal{L}, V)$ in general position, two distinct edges uv and xy are crossing if and only if the open segments uv and xy of pseudolines in $\mathcal{L}$ intersect.*

Proof. If uv and xy are crossing edges, then the open segments uv and xy must intersect; otherwise K along with uv and xy give an outerplanar drawing of K_4, which is impossible. If uv and xy are non-crossing while the open segments uv and xy intersect, then we can again obtain an outerplanar drawing of K_4 by re-connecting uv and xy in a sufficiently small neighbourhood of their unique intersection point – a contradiction. ◀

3 Obstructions for curve pseudo-visibility graphs

In this section, we discuss the obstructions mentioned in the introduction: the class $\mathcal{H}$ and the class of ordered holes. Ghosh [19] observed that these are obstructions for polygon visibility graphs, and related obstructions in the pseudo-visibility setting appear in the works of Abello and Kumar [2] and O'Rourke and Streinu [26].

Let u and v be two distinct vertices in an ordered graph $(G, \prec)$. If $u \prec v$, then a *crossing sequence* from u to v is a sequence of distinct edges $e_1, \ldots, e_k$ such that u is the smaller end of e_1, v is the larger end of e_k, and e_i crosses e_{i+1} for $1 \leqslant i < k$. The notion of a crossing sequence is invariant under rotation of $\prec$ as long as $u \prec v$. If $v \prec u$, then a *crossing sequence* from u to v is a crossing sequence from u to v in any rotation $\prec'$ of $\prec$ such that $u \prec' v$. These definitions should be thought of cyclically; whichever vertex is smaller, a crossing sequence from u to v begins at u and goes counterclockwise until it hits v (see Figure 5). If u and v are adjacent, then the edge uv is a crossing sequence from u to v and from v to u.

▶ **Lemma 3.1.** *If $(G, \prec)$ is an ordered graph with vertices $a \prec b \prec c \prec d$ and there are crossing sequences from a to c and from b to d, then there is a crossing sequence from a to d.*

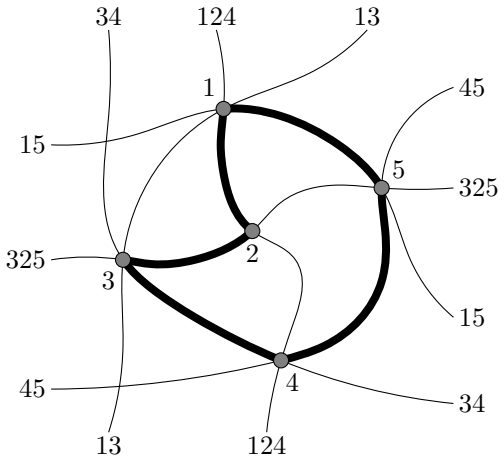


Figure 6 A pseudo-polygon with five articulation points: $1, 3, 4, 5$ are convex, 2 is concave.

Proof. Let $e_1, \ldots, e_k$ and $f_1, \ldots, f_t$ be crossing sequences from a to c and from b to d, respectively. Let e_i be the edge with the smallest index such that its larger end, say v, is greater than b in $\prec$. Let f_j be the edge with the largest index such that its smaller end is less than v in $\prec$. Then e_i crosses f_j and $e_1, \ldots, e_i, f_j, \ldots, f_t$ is a crossing sequence from a to d. ◀

The first family of obstructions, which we denote by $\mathcal{H}$, is defined as follows: $\mathcal{H}$ is the family of all ordered graphs containing two non-adjacent vertices u and v such that there exist a crossing sequence from u to v and a crossing sequence from v to u. See Figure 2 (left) for an illustration. The second family of obstructions is the family of ordered holes. An *ordered hole* is an ordered graph $(H, \prec)$ on vertex set $V(H) = \{c_1, \ldots, c_k\}$, where $k \geqslant 4$ and $c_1 \prec \cdots \prec c_k$, with edge set $E(H) = \{c_1c_2, \ldots, c_{k-1}c_k, c_kc_1\}$; see Figure 2 (middle).

▶ **Proposition 3.2.** *Every ordered curve pseudo-visibility graph is $\mathcal{H}$-free.*

▶ **Proposition 3.3.** *Every ordered curve pseudo-visibility graph is ordered-hole-free.*

We prove Proposition 3.2 later in this section, and the proof of Proposition 3.3 is in the full version. Before delving into the proof of the former, we show that we can test in polynomial time whether a given ordered graph is free of the considered obstructions.

▶ **Proposition 3.4.** *There is a polynomial-time algorithm which takes in an ordered graph $(G, \prec)$ and determines whether $(G, \prec)$ is $\mathcal{H}$-free.*

Proof. It suffices to test, for any two non-adjacent vertices u and v, whether $(G, \prec)$ has a crossing sequence from u to v. We assume that $u \prec v$ after possibly performing a rotation. We create a directed graph $\vec{H}$ with a vertex for each edge of G and with an arc from e to f for each pair of edges of G such that e crosses f. Then there is a crossing sequence from u to v in $(G, \prec)$ if and only if there is an edge e with smaller end u and an edge f with larger end v such that $\vec{H}$ has a directed path from e to f. ◀

▶ **Proposition 3.5.** *There is a polynomial-time algorithm which takes in an ordered graph $(G, \prec)$ and determines whether $(G, \prec)$ has an ordered hole.*

Proof. It suffices to test, for any two adjacent vertices $u \prec v$ of G, whether u and v are the first and last vertices of an ordered hole. This can be done by removing all vertices in a triangle with u and v and then testing for a directed path from u to v in the natural digraph. ◀

The proof of Proposition 3.2 requires some preparation. Let K be a pseudo-polygon on $\mathcal{L}$. A *segment* of K is a part of K that is contained in some pseudoline $L \in \mathcal{L}$ and connects two distinct intersection points of L with other pseudolines in $\mathcal{L}$. An *articulation point* of K is a point in K that joins two segments of K contained in distinct pseudolines in $\mathcal{L}$. Such an articulation point of K is *convex* if those two pseudolines extend to the exterior of K at p, and it is *concave* if they extend to the interior of K; see Figure 6. The following lemma was proven by Arroyo, Bensmail, and Richter [4]; we provide a proof for the reader's convenience.

▶ **Lemma 3.6.** *Every pseudo-polygon on $\mathcal{L}$ has at least three convex articulation points.*

Proof. Suppose otherwise, and choose a counterexample K with as few articulation points as possible. Since $\mathcal{L}$ is a pseudoline arrangement, K has at least three articulation points. Thus, we can choose consecutive articulation points p_1, p_2, and p_3 which occur on K in that order counterclockwise so that p_2 is concave and if any articulation point is convex, then p_3 is convex. Now, walk from p_1 towards p_2 along the pseudoline $L \in \mathcal{L}$ passing through p_1 and p_2, and continue walking on L beyond p_2 (through the interior of K, as p_2 is concave) until hitting K at a point $a \in K \cap L$. Let K' denote the pseudo-polygon formed by the segment p_2a of L and the part of K from a to p_2 counterclockwise. It follows that K' has at most two convex articulation points and has fewer articulation points than K, because at most one articulation point, a, is gained, and the articulation points p_2 and p_3 of K are lost. This is a contradiction, completing the proof. ◀

▶ **Lemma 3.7.** *Let K be a pseudo-polygon on $\mathcal{L}$. Let u and v be distinct points on K such that $\mathcal{L}$ contains a pseudoline L passing through u and v. If all articulation points of K other than possibly u and v are convex, then the segment uv of L is disjoint from the exterior of K.*

Proof. First, suppose that neither u nor v is a concave articulation point of K. Suppose for the sake of contradiction that the segment uv of L is not disjoint from the exterior of K, and let xy be a maximal subsegment of it with internal part contained in the exterior of K. Thus $x, y \in K$. The segment xy of L together with one of the parts of K between x and y forms a pseudo-polygon on $\mathcal{L}$ with interior contained in the exterior of K and with at most two convex articulation points: x and y. This contradicts Lemma 3.6.

Now, suppose that u is a concave articulation point of K while v is not. Let L' be a pseudoline containing one of the two segments of K incident to u. Follow L' from u in the other direction (towards the interior of K) until it hits K at some point x. Let K' be a pseudo-polygon formed by the segment ux of L' and the part of K between x and u that contains the point v. Thus x is a convex articulation point of K', u is no longer a concave articulation point of K', and every other articulation point of K that lies on K' remains convex on K'. Therefore, as we shown in the first case, the segment uv of L is disjoint from the exterior of K', so it is disjoint from the exterior of K.

The argument is analogous if v is a concave articulation point of K, except that when u is also a concave articulation point of K, then we apply the same argument as above to reduce to the case that only one of u, v is a concave articulation point of K. ◀

Proof of Proposition 3.2. Let $G = G_{\mathcal{L}}(K, V)$ be a curve pseudo-visibility graph and $\prec$ be a natural order of V on K. By Proposition 2.2, we can assume that $(\mathcal{L}, V)$ is in general position. For an edge $e = uv \in E(G)$, let ℓ_e denote the open segment uv of the pseudoline in $\mathcal{L}$ passing through u and v. Suppose that there are $u, v \in V$ with $u \prec v$ such that there are crossing sequences $e_1, \ldots, e_k$ from u to v and $f_1, \ldots, f_t$ from v to u. Choose the two crossing sequences so that $k + t$ is minimum. We need to show that uv is an edge of G.

By minimality and Lemmas 2.3 and 3.1, for $1 \leqslant i < j \leqslant k$, the segments ℓ_{e_i} and ℓ_{e_j} intersect if and only if $j = i + 1$, and likewise for the crossing sequence $f_1, \ldots, f_t$. Also by Lemma 2.3, each ℓ_{e_i} is disjoint from each ℓ_{f_j}. Therefore, by beginning at u and walking along ℓ_{e_1} until its unique intersection with ℓ_{e_2} is reached, then turning left and walking along ℓ_{e_2} until either v or its unique intersection with ℓ_{e_3} is reached, and so on, we can find an open curve $K_1 \subseteq \bigcup_{i=1}^{k} \ell_{e_i}$ with ends u and v. Likewise we can find an open curve $K_2 \subseteq \bigcup_{j=1}^{t} \ell_{f_j}$ with ends v and u. Let $K = K_1 \cup K_2 \cup \{u, v\}$. It follows that K is a pseudo-polygon on $\mathcal{L}$ and all articulation points of K except possibly u and v are convex. Therefore, by Lemma 3.7, the segment ℓ_{uv} is disjoint from the exterior of K, so $uv \in E(G)$. ◀

4 Partitioning into capped graphs

Recall that an ordered graph $(G, \prec)$ is *capped* if the following holds for any four vertices a, b, c, d with $a \prec b \prec c \prec d$: if $ac \in E(G)$ and $bd \in E(G)$, then $ad \in E(G)$. In contrast to previous notions defined in terms of $\prec$, this one is *not* invariant under rotation of $\prec$.

▶ **Lemma 4.1.** *If $u \prec v$ are two adjacent vertices of an $\mathcal{H}$-free ordered graph $(G, \prec)$ and $X = \{u, v\} \cup \{x \in V(G) \colon u \prec x \prec v \text{ and } ux, xv \in E(G)\}$, then $(G[X], \prec|_X)$ is a capped graph.*

Proof. If $a, b, c, d \in X$ and $ac, bd \in E(G)$, then ac, bd is a crossing sequence from a to d and du, va (da if $a = u$ or $d = v$) is a crossing sequence from d to a in $(G, \prec)$, so $ad \in E(G)$. ◀

Lemma 4.1 reduces computing the clique number of an $\mathcal{H}$-free ordered graph to computing the clique number of the capped graphs $(G[X], \prec|_X)$ for all adjacent pairs of vertices $u \prec v$. Thus, the following proposition allows us to conclude Theorem 1.3 from Theorem 1.4.

▶ **Proposition 4.2.** *There is a polynomial-time algorithm that takes in an $\mathcal{H}$-free ordered graph $(G, \prec)$ and partitions its set of vertices into three subsets V_1, V_2, and V_3 so that for each $i \in \{1, 2, 3\}$, the ordered graph $(G[V_i], \prec|_{V_i})$ is capped.*

The proof of Proposition 4.2 is technically complicated; see the full version of the paper. Below, we sketch the proof for the case that $(G, \prec)$ is an ordered polygon visibility graph. It is based on the "window partition" by Suri [34], which was used in a similar fashion to approximate chromatic variants of the well-known art gallery problem [8, 10].

Proof sketch for polygons. We write pq for the closed line segment connecting points p and q. Let $G = G(P, V)$ be a polygon visibility graph, where P is a polygon with vertex set V. Let $\prec$ be a natural ordering of G. Let x and y be the smallest and the largest vertex in $\prec$, respectively, so that xy is an edge of P and of G. Let $P_{xy} = (P \cup \operatorname{int} P) \setminus xy$, where $\operatorname{int} P$ is the interior of P. We construct a partition of $V(G)$ into three sets, which we express in terms of a colouring ϕ of $V(G)$ that uses three colours: red, green, and blue. First we describe a procedure that constructs a partition of P_{xy} into "windows"; these windows, as we will see, will be naturally arranged with a tree structure, and the root window will be "based" at xy.

To define the root window, we need to introduce the notion of "visibility from xy". We say that a point $p \in P_{xy}$ is *visible from* xy if p lies in the closed half-plane to the left of the line from y to x and there is a point $p' \in xy$ such that $pp' \subseteq P_{xy} \cup xy$. The *window* W_{xy} *based at* xy consists of all points $p \in P_{xy}$ that are visible from xy. It follows that W_{xy} is a connected subset of P_{xy}; see Figure 7 for an illustration. This set W_{xy} is the root of the constructed window partition tree.

The points in $P_{xy} \setminus W_{xy}$ form some number (possibly zero) of connected subsets of P_{xy}. It can be shown that each such set is of the form I_{ab} for some polygon I and edge ab of I, where a and b are on P and every point in the segment ab is visible from xy in P. Furthermore,

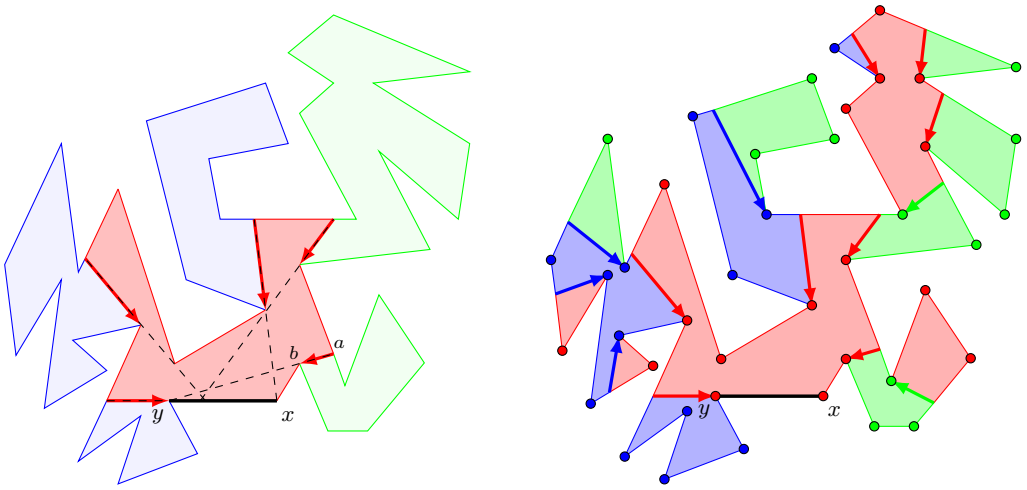


Figure 7 To the left: a polygon P with window W_{xy} based at xy in red, oriented lines $\overrightarrow{L_{ab}}$ depicted with red arrows and dashed lines, and the left/right-invisible sets I_{ab} in green/blue, respectively. To the right: the final window partition of P_{ab}.

at least one of the points a and b is a vertex of P and the line L_{ab} going through a and b intersects xy; we direct L_{ab} from a and b towards this intersection point to obtain an oriented line $\overrightarrow{L_{ab}}$. Given this description, we partition the invisible sets I_{ab} into two groups:

- I_{ab} is *left-invisible* if it is "towards the left side" of $\overrightarrow{L_{ab}}$;
- I_{ab} is *right-invisible* if it is "towards the right side" of $\overrightarrow{L_{ab}}$.

We do not give formal definitions, but refer to Figure 7.

Given this partition, it can be shown that there are no mutually visible points in two different left-invisible sets or two different right-invisible sets. Now, for each invisible set I_{ab}, we can recursively obtain a window partition of I_{ab} which is rooted at a window W_{ab} based at ab. The window partition of P_{xy} is then obtained by making each of these windows W_{ab} a *left-child* or a *right-child* of W_{xy} according to whether I_{ab} is left-invisible or right-invisible. The following observation summarizes this construction of the window partition: if two points in different windows W_1 and W_2 are mutually visible, then either W_1 and W_2 are in a parent-child relationship, or there is a window W such that one of W_1, W_2 is a left-child of W and the other is a right-child of W.

Now we show how to obtain the 3-colouring ϕ of the vertex set of $(G, \prec)$ such that each colour class induces a capped subgraph. The property above allows us to colour the windows by three colours (say, red, green, and blue) so that no two points in two different windows of the same colour are mutually visible. We colour the root window, say, by red. Then we extend this colouring on the remaining windows so that the children of each window W obtain a colour different from W and the left children of W are coloured with a different colour from the right children of W. This way every vertex of P other than x and y is coloured. We colour x and y arbitrarily; see Figure 7 (right).

To complete the proof, we need to show that the vertices of P in $W_{xy} \cup \{x, y\}$ induce a capped subgraph of $(G, \prec)$; for the other windows we can apply induction. It is well known that the related class of ordered terrain visibility graphs is capped [18, Lemma 1], but we give a proof sketch anyway, because that lemma does not apply directly.

Suppose there are four vertices $a, b, c, d \in W_{xy} \cup \{x, y\}$ such that $a \prec b \prec c \prec d$, $ac \in E(G)$, and $bd \in E(G)$ (it is possible that $a = x$ and/or $d = y$). By the definition of visibility from xy, all four points a, b, c, d are in the closed half-plane to the left of the line from y to x. Let

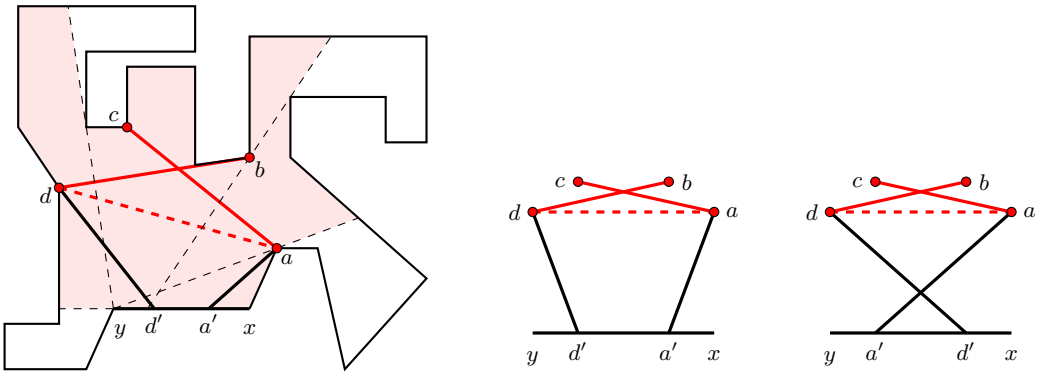


Figure 8 Possible relations between the segments $a'a, d'd, ac, bd, yx$.

$a', d' \in xy$ be such that the segments $a'a$ and $d'd$ are disjoint from the exterior of P; see Figure 8 for an illustration. Now, the five segments $a'a, d'd, ac, bd, yx$ divide the plane into a set $\mathcal{F}$ of faces, and exactly one of the faces in $\mathcal{F}$ is unbounded (the outer face). Now, to show that ad is disjoint from the exterior of P, it suffices to prove the following two claims.

- The polygon P is contained in the closure of the outer face of $\mathcal{F}$ and P does not cross (but may touch) any segment in the set $\{a'a, d'd, ac, bd\}$.
- The segment ad is disjoint from the interior of the outer face of $\mathcal{F}$.

The first claim is quite obvious; see the left side of Figure 8 for an illustration. The second claim can be proven by considering all the cases for how the segments $a'a$, $d'd$, ac, and bd can be placed with respect to each other. We leave the details to the reader. ◀

5 Colouring capped graphs

This section is devoted to the proof of Theorem 1.4 on colouring capped graphs. The proof relies on decompositions; a *decomposition* of an ordered graph $(G, \prec)$ is a collection of subgraphs such that every edge of G belongs to exactly one subgraph in the collection. For a graph G and a set $F \subseteq E(G)$, we write $G[F]$ and $G - F$ for the graph obtained from G by keeping/deleting (respectively) the edges in F.

▶ **Proposition 5.1.** *There is a polynomial-time algorithm which takes in a capped graph $(G, \prec)$ and returns its clique number ω and a decomposition of $(G, \prec)$ into $\omega - 1$ triangle-free capped graphs. If $(G, \prec)$ is additionally ordered-hole-free, then so is each graph in the decomposition.*

Proof. If G is triangle-free, then we return its clique number (1 or 2) and the decomposition consisting of $(G, \prec)$ itself. Thus assume that G is not triangle-free. We say that an edge $uv \in E(G)$ with $u \prec v$ is *triangle-crossed* if v belongs to a triangle with vertices x and y such that $x, y \preceq u$. Let F be the set of all edges that are not triangle-crossed (see Figure 9).

If a, b, c are vertices of G such that $a \prec b \prec c$, $ac, bc \in E(G)$, and ac is triangle-crossed, then bc is triangle-crossed. This implies that $(G[F], \prec)$ is capped and is ordered-hole-free if $(G, \prec)$ is ordered-hole-free. Furthermore, if vertices a, b, c with $a \prec b \prec c$ form a triangle in G, then bc is triangle-crossed. So $G[F]$ is triangle-free. Moreover, we have the following.

▷ Claim 5.2. The graph $(G - F, \prec)$ is capped and has clique number exactly one less than the clique number of $(G, \prec)$. Furthermore, if $(G, \prec)$ is ordered-hole-free, then so is $(G - F, \prec)$.

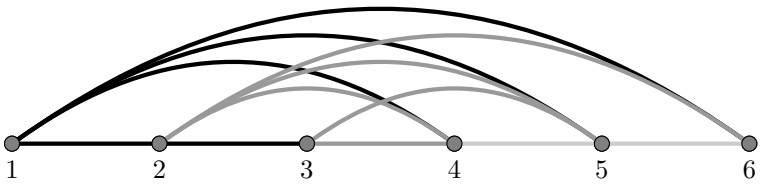


Figure 9 The decomposition from Proposition 5.1, where darker edges are removed first.

Proof. Let ω denote the clique number of $(G, \prec)$. If a, b, c are vertices of G such that $a \prec b \prec c$, $ab, ac \in E(G)$, and ab is triangle-crossed, then ac is triangle-crossed. So $(G - F, \prec)$ is capped and is ordered-hole-free if $(G, \prec)$ is. Furthermore, the clique number of $(G - F, \prec)$ is at least $\omega - 1$, because every edge of a clique in $(G, \prec)$ that is not incident to the smallest vertex of the clique is triangle-crossed. Now, suppose that $Q \subseteq V(G)$ is a clique in $G - F$, and let u and v be the two smallest vertices of Q, with $u \prec v$. Then v is in a triangle of G with vertices x and y such that $x \prec y \preceq u$. It follows that $(Q \setminus \{u\}) \cup \{x, y\}$ is a clique in G. This shows that the clique number of $(G - F, \prec)$ is at most $\omega - 1$, as desired. ◁

To conclude, the algorithm proceeds by continuing with $(G - F, \prec)$. The clique number is the number of subgraphs in the decomposition plus one. ◀

Proof of Theorem 1.4. Let $\omega \geqslant 2$ be the clique number of G, and let $\{(G_i, \prec)\}_{1 \leqslant i < \omega}$ be a decomposition of $(G, \prec)$ into $\omega - 1$ triangle-free capped subgraphs as in Proposition 5.1. Fix an index i with $1 \leqslant i < \omega$. If $(G, \prec)$ is ordered-hole-free, then let $F_i = \emptyset$, and otherwise let F_i be the set of edges of $(G_i, \prec)$ which are not crossed in $(G_i, \prec)$. An ordered graph is *outerplanar* if it has no crossing pair of edges.

▷ Claim 5.3. The ordered graph $(G[F_i], \prec)$ is outerplanar, and $(G_i - F_i, \prec)$ is both capped and ordered-hole-free.

Proof. We can assume that $(G, \prec)$ is not ordered-hole-free. That $(G[F_i], \prec)$ is outerplanar is clear from the definition of F_i. If a, b, c are vertices such that $a \prec b \prec c$, $ab, ac \in E(G_i)$, and ab is crossed, then ac is crossed. This implies that the ordered graph $(G_i - F_i, \prec)$ is capped and every ordered hole in it is an ordered hole in $(G_i, \prec)$. Suppose for the sake of contradiction that vertices $c_1 \prec \cdots \prec c_k$ induce an ordered hole in $(G_i - F_i, \prec)$ and thus in $(G_i, \prec)$. Since the edge $c_{k-2}c_{k-1}$ is crossed, there is an edge $xy \in E(G_i)$ with $x \prec c_{k-2} \prec y \prec c_{k-1}$. It follows that $x \prec c_1$, as $c_1, \ldots, c_k$ induce an ordered hole in $(G_i, \prec)$. We conclude that x, c_{k-1}, and c_k form a triangle in G_i. This contradiction shows that $(G_i - F_i, \prec)$ is ordered-hole-free. ◁

▷ Claim 5.4. There is a 4-colouring of $G_i - F_i$, which can be computed in polynomial time.

Proof. We just use the fact that $(G_i - F_i, \prec)$ is triangle-free, capped, and ordered-hole-free. For each component of $G_i - F_i$, we claim that any level of any breadth-first search tree which is rooted at the smallest vertex according to $\prec$ induces a bipartite subgraph. This suffices to complete the proof, as we can reuse colours at every second level.

Suppose for the sake of contradiction that p is the smallest vertex of a component and C is an induced odd cycle which is contained in a level. Since $(G_i - F_i, \prec)$ is triangle-free and ordered-hole-free, there are $ac, bd \in E(C)$ such that $a \prec b \prec c \prec d$. So $ad \in E(C)$, and none of the edges ac, bd, ad are crossing with any other edge of C. It follows that $V(C) \setminus \{a, d\}$ induces a path $bv_1 \cdots v_t c$ with $b \prec v_1 \prec \cdots \prec v_t \prec c$, for some positive integer t.

Let P be a shortest path from v_1 to p in $G_i - F_i$, and let v_1' be the vertex adjacent to v_1 in P. If $a \preceq v_1' \preceq d$ then we obtain a contradiction by finding a path which is shorter than P from either a or d to p, using the fact that $(G_i - F_i, \prec)$ is capped. Otherwise, if $v_1' \prec a$, then $\{v_1', v_1, \ldots, v_t, c\}$ contains a triangle or an ordered hole, which is a contradiction. In the final case that $d \prec v_1'$, the vertices b, v_1, v_1' form a triangle, which is again a contradiction. ◁

Let ϕ_i be a 4-colouring of $G_i - F_i$ from the last claim, for $1 \leqslant i < \omega$. Let $F = \bigcup_{i=1}^{\omega-1} F_i$. If $(G, \prec)$ is ordered-hole-free, then $F = \emptyset$ and the mapping $v \mapsto (\phi_1(v), \ldots, \phi_{\omega-1}(v))$ is a $4^{\omega-1}$-colouring of G. Otherwise, since every n-vertex outerplanar graph has at most $2n-3$ edges, every n-vertex subgraph of $G[F]$ has at most $(2n-3)(\omega-1)$ edges, for any $n \geqslant 2$. So every non-empty subgraph of $G[F]$ has a vertex of degree less than $4(\omega-1)$, and thus there exists a $4(\omega-1)$-colouring ψ of $G[F]$. Now, the mapping $v \mapsto (\phi_1(v), \ldots, \phi_{\omega-1}(v), \psi(v))$ is a $4^{\omega}(\omega-1)$-colouring of G. ◀

References

1. James Abello, Ömer Egecioglu, and Krishna Kumar. Visibility graphs of staircase polygons and the weak Bruhat order, I: from visibility graphs to maximal chains. *Discrete & Computational Geometry*, 14(3):331–358, 1995.
2. James Abello and Krishna Kumar. Visibility graphs and oriented matroids. *Discrete & Computational Geometry*, 28(4):449–465, 2002.
3. Safwa Ameer, Matt Gibson-Lopez, Erik Krohn, Sean Soderman, and Qing Wang. Terrain visibility graphs: persistence is not enough. In Sergio Cabello and Danny Z. Chen, editors, *36th International Symposium on Computational Geometry (SoCG 2020)*, volume 164 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 6:1–6:13. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Wadern, 2020.
4. Alan Arroyo, Julien Bensmail, and R. Bruce Richter. Extending drawings of graphs to arrangements of pseudolines. In Sergio Cabello and Danny Z. Chen, editors, *36th International Symposium on Computational Geometry (SoCG 2020)*, volume 164 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 9:1–9:14. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Wadern, 2020.
5. Edgar Asplund and Branko Grünbaum. On a coloring problem. *Mathematica Scandinavica*, 8:181–188, 1960.
6. David Avis and David Rappaport. Computing the largest empty convex subset of a set of points. In *Proceedings of the First Annual Symposium on Computational Geometry*, pages 161–167. Association for Computing Machinery, New York, 1985.
7. Maria Axenovich, Jonathan Rollin, and Torsten Ueckerdt. Chromatic number of ordered graphs with forbidden ordered subgraphs. *Combinatorica*, 38(5):1021–1043, 2018.
8. Andreas Bärtschi, Subir K. Ghosh, Matúš Mihalák, Thomas Tschager, and Peter Widmayer. Improved bounds for the conflict-free chromatic art gallery problem. In *Proceedings of the Thirtieth Annual Symposium on Computational Geometry*, pages 144–153. Association for Computing Machinery, New York, 2014.
9. Édouard Bonnet, Colin Geniet, Eun Jung Kim, Stéphan Thomassé, and Rémi Watrigant. Twin-width III: max independent set and coloring. arXiv, 2020. `arXiv:2007.14161`.
10. Onur Çağırıcı, Petr Hliněný, Subir K. Ghosh, and Bodhayan Roy. On conflict-free chromatic guarding of simple polygons. In Yingshu Li, Mihaela Cardei, and Yan Huang, editors, *Combinatorial Optimization and Applications*, volume 11949 of *Lecture Notes in Computer Science*, pages 601–612. Springer, Cham, 2019.
11. Onur Çağırıcı, Petr Hliněný, and Bodhayan Roy. On colourability of polygon visibility graphs. arXiv, 2019. `arXiv:1906.01904`.
12. Jean Cardinal and Udo Hoffmann. Recognition and complexity of point visibility graphs. *Discrete & Computational Geometry*, 57(1):164–178, 2017.

13 Daniele Catanzaro, Steven Chaplick, Stefan Felsner, Bjarni V. Halldórsson, Magnús M. Halldórsson, Thomas Hixon, and Juraj Stacho. Max point-tolerance graphs. *Discrete Applied Mathematics*, 216(1):84–97, 2017.

14 Maria Chudnovsky, Alex Scott, and Paul Seymour. Induced subgraphs of graphs with large chromatic number. V. Chandeliers and strings. arXiv, 2016. `arXiv:1609.00314`.

15 Alice M. Dean, William Evans, Ellen Gethner, Joshua D. Laison, Mohammad Ali Safari, and William T. Trotter. Bar k-visibility graphs: bounds on the number of edges, chromatic number, and thickness. In Patrick Healy and Nikola S. Nikolov, editors, *Graph Drawing*, volume 3843 of *Lecture Notes in Computer Science*, pages 73–82. Springer, Berlin, Heidelberg, 2006.

16 Stephan Eidenbenz and Christoph Stamm. Maximum clique and minimum clique partition in visibility graphs. In Jan van Leeuwen, Osamu Watanabe, Masami Hagiya, Peter D. Mosses, and Takayasu Ito, editors, *Theoretical Computer Science: Exploring New Frontiers of Theoretical Informatics*, pages 200–212. Springer, Berlin, Heidelberg, 2000.

17 Louis Esperet. *Graph Colorings, Flows and Perfect Matchings*. Habilitation thesis, Université Grenoble Alpes, 2017.

18 William S. Evans and Noushin Saeedi. On characterizing terrain visibility graphs. *Journal of Computational Geometry*, 6(1):108–141, 2015.

19 Subir K. Ghosh. On recognizing and characterizing visibility graphs of simple polygons. *Discrete & Computational Geometry*, 17(2):143–162, 1997.

20 Subir K. Ghosh and Partha P. Goswami. Unsolved problems in visibility graphs of points, segments, and polygons. *ACM Computing Surveys*, 46(2):Article 22, 2013.

21 Subir K. Ghosh, Thomas C. Shermer, Binay K. Bhattacharya, and Partha P. Goswami. Computing the maximum clique in the visibility graph of a simple polygon. *Journal of Discrete Algorithms*, 5(3):524–532, 2007.

22 András Gyárfás. On the chromatic number of multiple interval graphs and overlap graphs. *Discrete Mathematics*, 55(2):161–166, 1985.

23 Jan Kára, Attila Pór, and David R. Wood. On the chromatic number of the visibility graph of a set of points in the plane. *Discrete & Computational Geometry*, 34(3):497–506, 2005.

24 Friedrich Levi. Die Teilung der projektiven Ebene durch Gerade oder Pseudogerade. *Berichte über die Verhandlungen der Königlich-Sächsischen Gesellschaft der Wissenschaften zu Leipzig, Mathematisch-Physische Klasse*, 78:256–267, 1926.

25 Fabrizio Luccio, Silvia Mazzone, and Chak Kuen Wong. A note on visibility graphs. *Discrete Mathematics*, 64(2):209–219, 1987.

26 Joseph O'Rourke and Ileana Streinu. Vertex-edge pseudo-visibility graphs: characterization and recognition. In *Proceedings of the Thirteenth Annual Symposium on Computational Geometry*, pages 119–128. Association for Computing Machinery, New York, 1997.

27 János Pach and István Tomon. On the chromatic number of disjointness graphs of curves. *Journal of Combinatorial Theory, Series B*, 144:167–190, 2020.

28 Arkadiusz Pawlik, Jakub Kozik, Tomasz Krawczyk, Michał Lasoń, Piotr Micek, William T. Trotter, and Bartosz Walczak. Triangle-free intersection graphs of line segments with large chromatic number. *Journal of Combinatorial Theory, Series B*, 105:6–10, 2014.

29 Florian Pfender. Visibility graphs of point sets in the plane. *Discrete & Computational Geometry*, 39(1–3):455–459, 2008.

30 Alexandre Rok and Bartosz Walczak. Coloring curves that cross a fixed curve. *Discrete & Computational Geometry*, 61(4):830–851, 2019.

31 Alexandre Rok and Bartosz Walczak. Outerstring graphs are χ-bounded. *SIAM Journal on Discrete Mathematics*, 33(4):2181–2199, 2019.

32 Alex Scott and Paul Seymour. Induced subgraphs of graphs with large chromatic number. VI. Banana trees. *Journal of Combinatorial Theory, Series B*, 145:487–510, 2020.

33 Ileana Streinu. Non-stretchable pseudo-visibility graphs. *Computational Geometry*, 31(3):195–206, 2005.

34 Subhash Suri. A linear time algorithm for minimum link paths inside a simple polygon. *Computer Vision, Graphics, and Image Processing*, 35(1):99–110, 1986.

35 István Tomon. personal communication.

Computing Zigzag Persistence on Graphs in Near-Linear Time

Tamal K. Dey
Department of Computer Science, Purdue University, West Lafayette, IN, USA

Tao Hou
Department of Computer Science, Purdue University, West Lafayette, IN, USA

Abstract

Graphs model real-world circumstances in many applications where they may constantly change to capture the dynamic behavior of the phenomena. Topological persistence which provides a set of birth and death pairs for the topological features is one instrument for analyzing such changing graph data. However, standard persistent homology defined over a growing space cannot always capture such a dynamic process unless shrinking with deletions is also allowed. Hence, *zigzag persistence* which incorporates both insertions and deletions of simplices is more appropriate in such a setting. Unlike standard persistence which admits nearly linear-time algorithms for graphs, such results for the zigzag version improving the general $O(m^\omega)$ time complexity are not known, where $\omega < 2.37286$ is the matrix multiplication exponent. In this paper, we propose algorithms for zigzag persistence on graphs which run in near-linear time. Specifically, given a filtration with m additions and deletions on a graph with n vertices and edges, the algorithm for 0-dimension runs in $O(m \log^2 n + m \log m)$ time and the algorithm for 1-dimension runs in $O(m \log^4 n)$ time. The algorithm for 0-dimension draws upon another algorithm designed originally for pairing critical points of Morse functions on 2-manifolds. The algorithm for 1-dimension pairs a negative edge with the *earliest* positive edge so that a 1-cycle containing both edges resides in all intermediate graphs. Both algorithms achieve the claimed time complexity via dynamic graph data structures proposed by Holm et al. In the end, using Alexander duality, we extend the algorithm for 0-dimension to compute the $(p-1)$-dimensional zigzag persistence for $\mathbb{R}^p$-embedded complexes in $O(m \log^2 n + m \log m + n \log n)$ time.

2012 ACM Subject Classification Theory of computation → Computational geometry; Mathematics of computing → Algebraic topology

Keywords and phrases persistent homology, zigzag persistence, graph filtration, dynamic networks

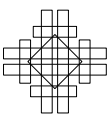
Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.30

Related Version *Full Version*: `https://arxiv.org/abs/2103.07353` [7]

Funding This research is supported by NSF grants CCF 1839252 and 2049010.

1 Introduction

Graphs appear in many applications as abstraction of real-world phenomena, where vertices represent certain objects and edges represent their relations. Rather than being stationary, graph data obtained in applications usually change with respect to some parameter such as time. A summary of these changes in a quantifiable manner can help gain insight into the data. Persistent homology [3, 10] is a suitable tool for this goal because it quantifies the life span of topological features as the graph changes. One drawback of using standard non-zigzag persistence [10] is that it only allows addition of vertices and edges during the change, whereas deletion may also happen in practice. For example, many complex systems such as social networks, food webs, or disease spreading are modeled by the so-called "dynamic networks" [13, 14, 19], where vertices and edges can appear and disappear at different time. A variant of the standard persistence called *zigzag persistence* [3] is thus a more natural tool in such scenarios because simplices can be both added and deleted. Given



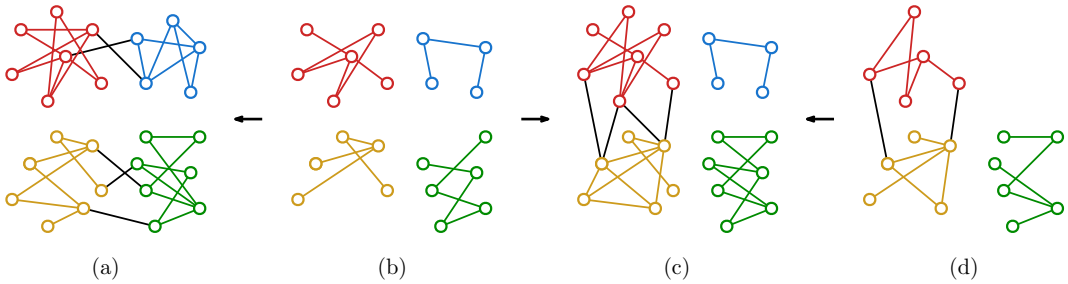



Figure 1 A sequence of graphs with four prominent clusters each colored differently. Black edges connect different clusters and forward (resp. backward) arrows indicate additions (resp. deletions) of vertices and edges. From (a) to (b), two clusters split; from (b) to (c), two clusters merge; from (c) to (d), one cluster disappears.

a sequence of graphs possibly with additions and deletions (formally called a *zigzag filtration*), zigzag persistence produces a set of intervals termed as *zigzag barcode* in which each interval registers the birth and death time of a homological feature. Figure 1 gives an example of a graph sequence in which clusters may split (birth of 0-dimensional features) or vanish/merge (death of 0-dimensional features). Moreover, addition of edges within the clusters creates 1-dimensional cycles and deletion of edges makes some cycles disappear. These births and deaths are captured by zigzag persistence.

Algorithms for both zigzag and non-zigzag persistence have a general-case time complexity of $O(m^\omega)$ [4, 10, 15, 16], where m is the length of the input filtration and $\omega < 2.37286$ is the matrix multiplication exponent [2]. For the special case of graph filtrations, it is well known that non-zigzag persistence can be computed in $O(m\,\alpha(m))$ time, where $\alpha(m)$ is the inverse Ackermann's function that is almost constant for all practical purposes [5]. However, analogous faster algorithms for zigzag persistence on graphs are not known. In this paper, we present algorithms for zigzag persistence on graphs with near-linear time complexity. In particular, given a zigzag filtration of length m for a graph with n vertices and edges, our algorithm for 0-dimension runs in $O(m \log^2 n + m \log m)$ time, and our algorithm for 1-dimension runs in $O(m \log^4 n)$ time. Observe that the algorithm for 0-dimension works for arbitrary complexes by restricting to the 1-skeletons.

The difficulty in designing faster zigzag persistence algorithms for the special case of graphs lies in the deletion of vertices and edges. For example, besides merging into bigger ones, connected components can also split into smaller ones because of edge deletion. Therefore, one cannot simply kill the younger component during merging as in standard persistence [10], but rather has to pair the *merge* and *departure* events with the *split* and *entrance* events (see Sections 3 for details). Similarly, in dimension one, deletion of edges may kill 1-cycles so that one has to properly pair the creation and destruction of 1-cycles, instead of simply treating all 1-dimensional intervals as infinite ones.

Our solutions are as follows: in dimension zero, we find that the $O(n \log n)$ algorithm by Agarwal et al. [1] originally designed for pairing critical points of Morse functions on 2-manifolds can be utilized in our scenario. We formally prove the correctness of applying the algorithm and use a *dynamic connectivity* data structure [12] to achieve the claimed complexity. In dimension one, we observe that a positive and a negative edge can be paired by finding the *earliest* 1-cycle containing both edges which resides in all intermediate graphs. We further reduce the pairing to finding the *max edge-weight* of a path in a minimum spanning forest. Utilizing a data structure for *dynamic minimum spanning forest* [12], we achieve the claimed time complexity. Section 4 details this algorithm.

Using Alexander duality, we also extend the algorithm for 0-dimension to compute $(p-1)$-dimensional zigzag for $\mathbb{R}^p$-embedded complexes. The connection between these two cases for non-zigzag persistence is well known [9, 18], and the challenge comes in adopting this duality to the zigzag setting while maintaining an efficient time budget. With the help of a *dual filtration* and an observation about faster void boundary reconstruction for $(p-1)$-*connected* complexes [8], we achieve a time complexity of $O(m \log^2 n + m \log m + n \log n)$.

All omitted proofs in this version appear in the full version [7].

2 Preliminaries

A *zigzag module* (or *module* for short) is a sequence of vector spaces

$$\mathcal{M} : V_0 \xleftrightarrow{\psi_0} V_1 \xleftrightarrow{\psi_1} \cdots \xleftrightarrow{\psi_{m-1}} V_m$$

in which each ψ_i is either a forward linear map $\psi_i : V_i \to V_{i+1}$ or a backward linear map $\psi_i : V_i \leftarrow V_{i+1}$. We assume vector spaces are over field $\mathbb{Z}_2$ in this paper. A module $\mathcal{S}$ of the form

$$\mathcal{S} : W_0 \xleftrightarrow{\phi_0} W_1 \xleftrightarrow{\phi_1} \cdots \xleftrightarrow{\phi_{m-1}} W_m$$

is called a *submodule* of $\mathcal{M}$ if each W_i is a subspace of V_i and each ϕ_i is the restriction of ψ_i. For an interval $[b, d] \subseteq [0, m]$, $\mathcal{S}$ is called an *interval submodule* of $\mathcal{M}$ over $[b, d]$ if W_i is one-dimensional for $i \in [b, d]$ and is trivial for $i \notin [b, d]$, and ϕ_i is an isomorphism for $i \in [b, d-1]$. It is well known [3] that $\mathcal{M}$ admits an *interval decomposition* $\mathcal{M} = \bigoplus_{\alpha \in \mathcal{A}} \mathcal{I}^{[b_\alpha, d_\alpha]}$ which is a direct sum of interval submodules of $\mathcal{M}$. The (multi-)set of intervals $\{[b_\alpha, d_\alpha] \,|\, \alpha \in \mathcal{A}\}$ is called the *zigzag barcode* (or *barcode* for short) of $\mathcal{M}$ and is denoted as $\mathsf{Pers}(\mathcal{M})$. Each interval in a zigzag barcode is called a *persistence interval*.

In this paper, we mainly focus on a special type of zigzag modules:

▶ **Definition 1** (Elementary zigzag module)**.** *A zigzag module is called* ***elementary*** *if it starts with the trivial vector space and all linear maps in the module are of the three forms:* (i) *an isomorphism;* (ii) *an injection with rank 1 cokernel;* (iii) *a surjection with rank 1 kernel.*

A *zigzag filtration* (or *filtration* for short) is a sequence of simplicial complexes

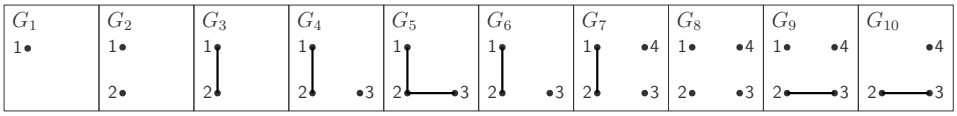
$$\mathcal{F} : K_0 \xleftrightarrow{\sigma_0} K_1 \xleftrightarrow{\sigma_1} \cdots \xleftrightarrow{\sigma_{m-1}} K_m$$

in which each $K_i \xleftrightarrow{\sigma_i} K_{i+1}$ is either a forward inclusion $K_i \hookrightarrow K_{i+1}$ with a single simplex σ_i added, or a backward inclusion $K_i \hookleftarrow K_{i+1}$ with a single σ_i deleted. When the σ_i's are not explicitly used, we drop them and simply denote $\mathcal{F}$ as $\mathcal{F} : K_0 \leftrightarrow K_1 \leftrightarrow \cdots \leftrightarrow K_m$. For computational purposes, we sometimes assume that a filtration starts with the empty complex, i.e., $K_0 = \varnothing$ in $\mathcal{F}$. Throughout the paper, we also assume that each K_i in $\mathcal{F}$ is a subcomplex of a fixed complex K; such a K, when not given, can be constructed by taking the union of every K_i in $\mathcal{F}$. In this case, we call $\mathcal{F}$ a filtration *of* K.

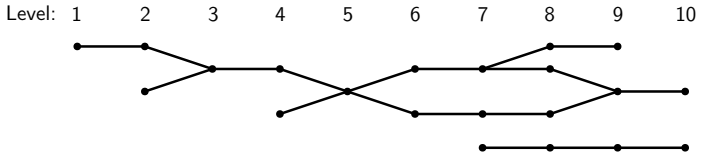
Applying the p-th homology with $\mathbb{Z}_2$ coefficients on $\mathcal{F}$, we derive the *p-th zigzag module of* $\mathcal{F}$

$$\mathsf{H}_p(\mathcal{F}) : \mathsf{H}_p(K_0) \xleftrightarrow{\varphi_0^p} \mathsf{H}_p(K_1) \xleftrightarrow{\varphi_1^p} \cdots \xleftrightarrow{\varphi_{m-1}^p} \mathsf{H}_p(K_m)$$

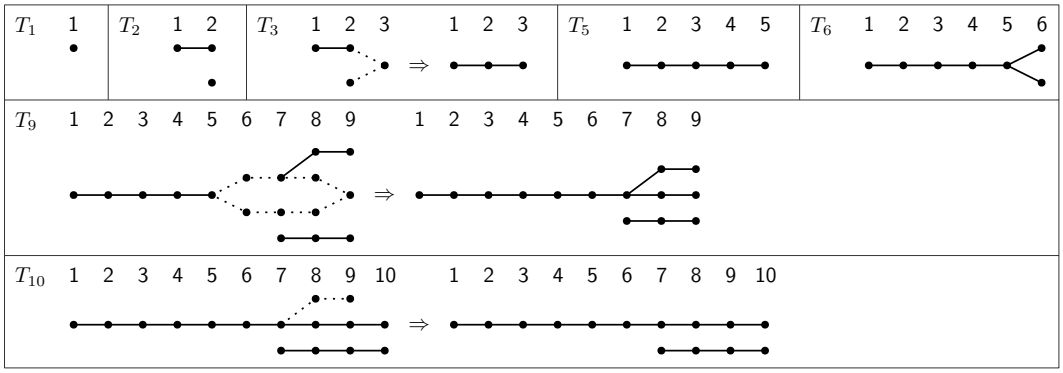
in which each φ_i^p is the linear map induced by the inclusion. In this paper, whenever $\mathcal{F}$ is used to denote a filtration, we use φ_i^p to denote a linear map in the module $\mathsf{H}_p(\mathcal{F})$. Note that $\mathsf{H}_p(\mathcal{F})$ is an elementary module if $\mathcal{F}$ starts with an empty complex. Specifically, we call $\mathsf{Pers}(\mathsf{H}_p(\mathcal{F}))$ the *p-th zigzag barcode* of $\mathcal{F}$.



(a) A zigzag filtration of graphs with 0-th barcode $\{[2,2],[4,4],[6,8],[8,9],[7,10],[1,10]\}$.



(b) The barcode graph for the filtration shown in Figure 2a.



(c) Barcode forests constructed in Algorithm 1 for the barcode graph in Figure 2b. For brevity, some forests are skipped. The horizontally arranged labels indicate the levels.

Figure 2 Examples of a zigzag filtration, a barcode graph, and barcode forests.

3 Zero-dimensional zigzag persistence

We present our algorithm for 0-th zigzag persistence[1] in this section. The input is assumed to be on graphs but note that our algorithm can be applied to any complex by restricting to its 1-skeleton. We first define the barcode graph of a zigzag filtration which is a construct that our algorithm implicitly works on. In a barcode graph, nodes correspond to connected components of graphs in the filtration and edges encode the mapping between the components:

▶ **Definition 2** (Barcode graph). *For a graph G and a zigzag filtration $\mathcal{F} : G_0 \leftrightarrow G_1 \leftrightarrow \cdots \leftrightarrow G_m$ of G, the **barcode graph** $\mathbb{G}_{\mathrm{B}}(\mathcal{F})$ of $\mathcal{F}$ is a graph whose vertices (preferably called **nodes**) are associated with a **level** and whose edges connect nodes only at adjacent levels. The graph $\mathbb{G}_{\mathrm{B}}(\mathcal{F})$ is constructively described as follows:*

- *For each G_i in $\mathcal{F}$ and each connected component of G_i, there is a node in $\mathbb{G}_{\mathrm{B}}(\mathcal{F})$ at level i corresponding to this component; this node is also called a **level-i node**.*
- *For each inclusion $G_i \leftrightarrow G_{i+1}$ in $\mathcal{F}$, if it is forward, then there is an edge connecting a level-i node v_i to a level-$(i+1)$ node v_{i+1} if and only if the component of v_i maps to the component of v_{i+1} by the inclusion. Similarly, if the inclusion is backward, then v_i connects to v_{i+1} by an edge iff the component of v_{i+1} maps to the component of v_i.*

*For two nodes at different levels in $\mathbb{G}_{\mathrm{B}}(\mathcal{F})$, the node at the higher (resp. lower) level is said to be **higher** (resp. **lower**) than the other.*

[1] For brevity, henceforth we call p-dimensional zigzag persistence as *p-th* zigzag persistence.

▶ Remark 3. Note that some works [6, 14] also have used similar notions of barcode graphs.

Figure 2a and 2b give an example of a zigzag filtration and its barcode graph. Note that a barcode graph is of size $O(mn)$, where m is the length of $\mathcal{F}$ and n is the number of vertices and edges of G. Although we present our algorithm (Algorithm 1) by first building the barcode graph, the implementation does not do so explicitly, allowing us to achieve the claimed time complexity; see Section 3.1 for the implementation details. Introducing barcode graphs helps us justify the algorithm, and more importantly, points to the fact that the algorithm can be applied whenever such a barcode graph can be built.

Algorithm 1 Algorithm for 0-th zigzag persistence.

Given a graph G and a zigzag filtration $\mathcal{F} : \varnothing = G_0 \leftrightarrow G_1 \leftrightarrow \cdots \leftrightarrow G_m$ of G, we first build the barcode graph $\mathbb{G}_{\mathrm{B}}(\mathcal{F})$, and then apply the pairing algorithm described in [1] on $\mathbb{G}_{\mathrm{B}}(\mathcal{F})$ to compute $\mathsf{Pers}(\mathsf{H}_0(\mathcal{F}))$. For a better understanding, we rephrase this algorithm which originally works on Reeb graphs:

The algorithm iterates for $i = 0, \ldots, m-1$ and maintains a **barcode forest** T_i, whose leaves have a one-to-one correspondence to level-i nodes of $\mathbb{G}_{\mathrm{B}}(\mathcal{F})$. Like the barcode graph, each tree node in a barcode forest is associated with a level and each tree edge connects nodes at adjacent levels. For each tree in a barcode forest, the lowest node is the root. Initially, T_0 is empty; then, the algorithm builds T_{i+1} from T_i in the i-th iteration. Intervals for $\mathsf{Pers}(\mathsf{H}_0(\mathcal{F}))$ are produced while updating the barcode forest. (Figure 2c illustrates such updates.)

Specifically, the i-th iteration proceeds as follows: first, T_{i+1} is formed by copying the level-$(i+1)$ nodes of $\mathbb{G}_{\mathrm{B}}(\mathcal{F})$ and their connections to the level-i nodes, into T_i; the copying is possible because leaves of T_i and level-i nodes of $\mathbb{G}_{\mathrm{B}}(\mathcal{F})$ have a one-to-one correspondence; see transitions from T_5 to T_6 and from T_9 to T_{10} in Figure 2c. We further change T_{i+1} under the following events:

Entrance: One level-$(i+1)$ node in T_{i+1}, said to be **entering**, does not connect to any level-i node.

Split: One level-i node in T_{i+1}, said to be **splitting**, connects to two different level-$(i+1)$ nodes. For the two events so far, no changes need to be made on T_{i+1}.

Departure: One level-i node u in T_{i+1}, said to be **departing**, does not connect to any level-$(i+1)$ node. If u has splitting ancestors (i.e., ancestors which are also splitting nodes), add an interval $[j+1, i]$ to $\mathsf{Pers}(\mathsf{H}_0(\mathcal{F}))$, where j is the level of the highest splitting ancestor v of u; otherwise, add an interval $[j, i]$ to $\mathsf{Pers}(\mathsf{H}_0(\mathcal{F}))$, where j is the level of the root v of u. We then delete the path from v to u in T_{i+1}.

Merge: Two different level-i nodes u_1, u_2 in T_{i+1} connect to the same level-$(i+1)$ node. Tentatively, T_{i+1} may now contain a loop and is not a tree. If u_1, u_2 are in different trees in T_i, add an interval $[j, i]$ to $\mathsf{Pers}(\mathsf{H}_0(\mathcal{F}))$, where j is the level of the higher root of u_1, u_2 in T_i; otherwise, add an interval $[j+1, i]$ to $\mathsf{Pers}(\mathsf{H}_0(\mathcal{F}))$, where j is the level of the highest common ancestor of u_1, u_2 in T_i. We then glue the two paths from u_1 and u_2 to their level-j ancestors in T_{i+1}, after which T_{i+1} is guaranteed to be a tree.

No-change: If none of the above events happen, no changes are made on T_{i+1}.

At the end, for each root in T_m at a level j, add an interval $[j, m]$ to $\mathsf{Pers}(\mathsf{H}_0(\mathcal{F}))$, and for each splitting node in T_m at a level j, add an interval $[j+1, m]$ to $\mathsf{Pers}(\mathsf{H}_0(\mathcal{F}))$.

▶ Remark 4. The justification of Algorithm 1 is given in Section 3.2.

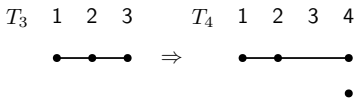


Figure 3 For the example in Figure 2, to form T_4, our implementation only adds a level-4 entering node, whereas the leaf in T_3 is not touched. Since the level of a leaf always equals the index of the barcode forest, the leaf at level 3 in T_3 automatically becomes a leaf at level 4 in T_4.

Figure 2c gives examples of barcode forests constructed by Algorithm 1 for the barcode graph shown in Figure 2b, where T_1 and T_2 introduce entering nodes, T_6 introduces a splitting node, and T_{10} introduces a departing node. In T_{10}, the departure event happens and the dotted path is deleted, producing an interval $[8, 9]$. In T_3 and T_9, the merge event happens and the dotted paths are glued together, producing intervals $[2, 2]$ and $[6, 8]$. Note that the glued level-i nodes are in different trees in T_3 and are in the same tree in T_9.

3.1 Implementation

Inserting (resp. deleting) a vertex in $\mathcal{F}$ simply corresponds to the entrance (resp. departure) event, whereas inserting (resp. deleting) an edge corresponds to the merge (resp. split) event only when connected components in the graph merge (resp. split). To keep track of the connectivity of vertices for each G_i, we use a *dynamic connectivity* data structure by Holm et al. [12], which we denote as $\mathbb{D}$. The data structure $\mathbb{D}$ dynamically updates the spanning forest of G_i when edges are inserted or deleted, and the connected component of a vertex of G_i can be identified by the tree in the spanning forest containing the vertex. The total time for querying and updating $\mathbb{D}$ is $O(m \log^2 n)$ [12], where m is the length of $\mathcal{F}$ and n is the number of vertices and edges of G. As mentioned, we do not explicitly build a barcode graph, but instead maintain a key-value map from connected components of G_i (i.e., identifiers of trees in $\mathbb{D}$'s spanning forest) to leaves of T_i. We update T_i to form T_{i+1} according to the changes of the connected components from G_i to G_{i+1}. We also only record the level of a non-leaf node in a barcode forest T_i because the level of a leaf always equals i. Note that at iteration i, a leaf in T_i may uniquely connect to a single leaf in T_{i+1}. In this case, we simply let the leaf in T_i automatically become a leaf in T_{i+1}; see Figure 3. The size of a barcode forest is then $O(m)$. As in [1], using the *mergeable trees* data structure [11], the traversal and updates of the barcode forest can be done in $O(m \log m)$ time. Therefore, the time complexity of Algorithm 1 is $O(m \log^2 n + m \log m)$. See the full version of the paper [7] for more details on the implementation of Algorithm 1.

3.2 Justification

In this subsection, we justify the correctness of Algorithm 1. For each entering node u in a T_i of Algorithm 1, there must be a single node in $\mathbb{G}_{\mathrm{B}}(\mathcal{F})$ at the level of u with the same property. So we also have entering nodes in $\mathbb{G}_{\mathrm{B}}(\mathcal{F})$. Splitting and departing nodes in $\mathbb{G}_{\mathrm{B}}(\mathcal{F})$ can be similarly defined.

We first prepare some standard notions and facts in zigzag persistence (Definition 5 and 7, Proposition 9) that help with our proofs. Some notions also appear in previous works in different forms; see, e.g., [15].

▶ **Definition 5** (Representatives). *Let $\mathcal{M} : V_0 \xleftrightarrow{\psi_0} \cdots \xleftrightarrow{\psi_{m-1}} V_m$ be an elementary zigzag module and $[s, t] \subseteq [1, m]$ be an interval. An indexed set $\{\alpha_i \in V_i \mid i \in [s, t]\}$ is called a set of **partial representatives** for $[s, t]$ if for each $i \in [s, t-1]$, $\alpha_i \mapsto \alpha_{i+1}$ or $\alpha_i \mapsfrom \alpha_{i+1}$ by ψ_i; it is called a set of **representatives** for $[s, t]$ if the following additional conditions are satisfied:*

1. *If $\psi_{s-1} : V_{s-1} \to V_s$ is forward with non-trivial cokernel, then α_s is not in $\mathsf{img}(\psi_{s-1})$; if $\psi_{s-1} : V_{s-1} \leftarrow V_s$ is backward with non-trivial kernel, then α_s is the non-zero element in $\mathsf{ker}(\psi_{s-1})$.*
2. *If $t < m$ and $\psi_t : V_t \leftarrow V_{t+1}$ is backward with non-trivial cokernel, then α_t is not in $\mathsf{img}(\psi_t)$; if $t < m$ and $\psi_t : V_t \to V_{t+1}$ is forward with non-trivial kernel, then α_t is the non-zero element in $\mathsf{ker}(\psi_t)$.*

*Specifically, when $\mathcal{M} := \mathsf{H}_p(\mathcal{F})$ for a zigzag filtration $\mathcal{F}$, we use terms p-**representatives** and **partial** p-**representatives** to emphasize the dimension p.*

▶ Remark 6. Let $\mathcal{F}$ be the filtration given in Figure 2a, and let α_8, α_9 be the sum of the component containing vertex 1 and the component containing vertex 2 in G_8 and G_9. Then, $\{\alpha_8, \alpha_9\}$ is a set of 0-representatives for the interval $[8, 9] \in \mathsf{Pers}(\mathsf{H}_0(\mathcal{F}))$.

▶ **Definition 7** (Positive/negative indices). *Let $\mathcal{M} : V_0 \xleftrightarrow{\psi_0} \cdots \xleftrightarrow{\psi_{m-1}} V_m$ be an elementary zigzag module. The set of **positive indices** of $\mathcal{M}$, denoted $\mathsf{P}(\mathcal{M})$, and the set of **negative indices** of $\mathcal{M}$, denoted $\mathsf{N}(\mathcal{M})$, are constructed as follows: for each forward $\psi_i : V_i \to V_{i+1}$, if ψ_i is an injection with non-trivial cokernel, add $i+1$ to $\mathsf{P}(\mathcal{M})$; if ψ_i is a surjection with non-trivial kernel, add i to $\mathsf{N}(\mathcal{M})$. Furthermore, for each backward $\psi_i : V_i \leftarrow V_{i+1}$, if ψ_i is an injection with non-trivial cokernel, add i to $\mathsf{N}(\mathcal{M})$; if ψ_i is a surjection with non-trivial kernel, add $i+1$ to $\mathsf{P}(\mathcal{M})$. Finally, add $\mathsf{rank}\, V_m$ copies of m to $\mathsf{N}(\mathcal{M})$.*

▶ Remark 8. For each $\psi_i : V_i \leftrightarrow V_{i+1}$ in Definition 7, if $i+1 \in \mathsf{P}(\mathcal{M})$, then $i \notin \mathsf{N}(\mathcal{M})$; similarly, if $i \in \mathsf{N}(\mathcal{M})$, then $i+1 \notin \mathsf{P}(\mathcal{M})$. Furthermore, if ψ_i is an isomorphism, then $i \notin \mathsf{N}(\mathcal{M})$ and $i+1 \notin \mathsf{P}(\mathcal{M})$.

Note that $\mathsf{N}(\mathcal{M})$ in Definition 7 is in fact a multi-set; calling it a set should not cause any confusion in this paper though. Also note that $|\mathsf{P}(\mathcal{M})| = |\mathsf{N}(\mathcal{M})|$, and every index in $\mathsf{P}(\mathcal{M})$ (resp. $\mathsf{N}(\mathcal{M})$) is the start (resp. end) of an interval in $\mathsf{Pers}(\mathcal{M})$. This explains why we add $\mathsf{rank}\, V_m$ copies of m to $\mathsf{N}(\mathcal{M})$ because there are always $\mathsf{rank}\, V_m$ number of intervals ending with m in $\mathsf{Pers}(\mathcal{M})$; see the example in Figure 2a where $\mathsf{rank}\, \mathsf{H}_0(G_{10}) = 2$.

▶ **Proposition 9.** *Let $\mathcal{M}$ be an elementary zigzag module and $\pi : \mathsf{P}(\mathcal{M}) \to \mathsf{N}(\mathcal{M})$ be a bijection. If every $b \in \mathsf{P}(\mathcal{M})$ satisfies that $b \le \pi(b)$ and the interval $[b, \pi(b)]$ has a set of representatives, then $\mathsf{Pers}(\mathcal{M}) = \{[b, \pi(b)] \mid b \in \mathsf{P}(\mathcal{M})\}$.*

Now we present several propositions leading to our conclusion (Theorem 14). Specifically, Proposition 10 states that a certain path in $\mathbb{G}_{\mathrm{B}}(\mathcal{F})$ induces a set of partial 0-representatives. Proposition 11 lists some invariants of Algorithm 1. Proposition 10 and 11 support the proof of Proposition 13, which together with Proposition 9 implies Theorem 14.

From now on, G and $\mathcal{F}$ always denote the input to Algorithm 1. Since each node in a barcode graph represents a connected component, we also interpret nodes in a barcode graph as 0-th homology classes throughout the paper. Moreover, a path in a barcode graph from a node v to a node u is said to be *within level j and i* if for each node on the path, its level ℓ satisfies $j \le \ell \le i$; we denote such a path as $(v \rightsquigarrow u)_{[j,i]}$.

▶ **Proposition 10.** *Let v be a level-j node and u be a level-i node in $\mathbb{G}_{\mathrm{B}}(\mathcal{F})$ such that $j < i$ and there is a path $(v \rightsquigarrow u)_{[j,i]}$ in $\mathbb{G}_{\mathrm{B}}(\mathcal{F})$. Then, there is a set of partial 0-representatives $\{\alpha_k \in \mathsf{H}_0(G_k) \mid k \in [j,i]\}$ for the interval $[j,i]$ with $\alpha_j = v$ and $\alpha_i = u$.*

Proof. We can assume that $(v \rightsquigarrow u)_{[j,i]}$ is a simple path because if it were not we could always find one. For each $k \in [j+1, i-1]$, let $w_1, \ldots, w_r$ be all the level-k nodes on $(v \rightsquigarrow u)_{[j,i]}$ whose adjacent nodes on $(v \rightsquigarrow u)_{[j,i]}$ are at different levels. Then, let $\alpha_k = \sum_{\ell=1}^{r} w_\ell$. Also, let

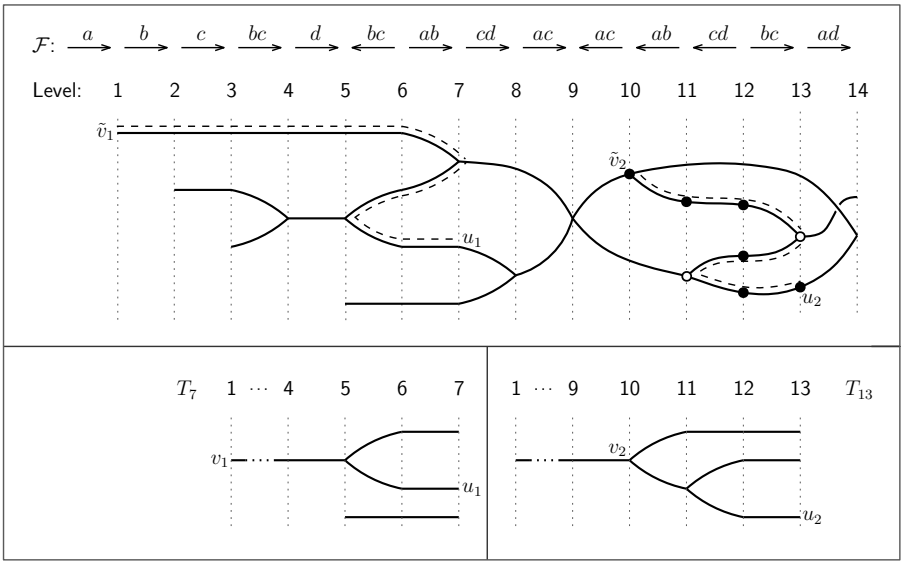


Figure 4 Illustration of invariants of Proposition 11. The top part contains a barcode graph with its filtration given (a, b, c, and d are vertices of the complex). The bottom contains two barcode forests.

$\alpha_j = v$ and $\alpha_i = u$. It can be verified that $\{\alpha_k \mid k \in [j,i]\}$ is a set of partial 0-representatives for $[j,i]$. See Figure 4 for an example of a simple path $(\tilde{v}_2 \rightsquigarrow u_2)_{[10,13]}$ (the dashed one) in a barcode graph, where the solid nodes contribute to the induced partial 0-representatives and the hollow nodes are excluded. ◀

For an i with $0 \le i \le m$, we define the *prefix* $\mathcal{F}^i$ of $\mathcal{F}$ as the filtration $\mathcal{F}^i : G_0 \leftrightarrow \cdots \leftrightarrow G_i$ and observe that $\mathbb{G}_{\mathrm{B}}(\mathcal{F}^i)$ is the subgraph of $\mathbb{G}_{\mathrm{B}}(\mathcal{F})$ induced by nodes at levels less than or equal to i. We call level-i nodes of $\mathbb{G}_{\mathrm{B}}(\mathcal{F}^i)$ as *leaves* and do not distinguish leaves in T_i and $\mathbb{G}_{\mathrm{B}}(\mathcal{F}^i)$ because they bijectively map to each other. It should be clear from the context though which graph or forest a particular leaf is in.

▶ **Proposition 11.** *For each $i = 0, \ldots, m$, Algorithm 1 maintains the following invariants:*

1. *There is a bijection η from trees in T_i to connected components in $\mathbb{G}_{\mathrm{B}}(\mathcal{F}^i)$ containing leaves such that a leaf u is in a tree Υ of T_i if and only if u is in $\eta(\Upsilon)$.*
2. *For each leaf u in T_i and each ancestor of u at a level j, there is a path $(\tilde{v} \rightsquigarrow u)_{[j,i]}$ in $\mathbb{G}_{\mathrm{B}}(\mathcal{F})$ where $\tilde{v}$ is a level-j node.*
3. *For each leaf u in T_i and each splitting ancestor of u at a level j, let $\tilde{v}$ be the unique level-j splitting node in $\mathbb{G}_{\mathrm{B}}(\mathcal{F})$. Then, there is a path $(\tilde{v} \rightsquigarrow u)_{[j,i]}$ in $\mathbb{G}_{\mathrm{B}}(\mathcal{F})$.*

▶ Remark 12. See Figure 4 for examples of invariant 2 and 3. In the figure, v_1 is a level-1 non-splitting ancestor of u_1 in T_7 and $\tilde{v}_1$ is a level-1 node in the barcode graph; v_2 is a level-10 splitting ancestor of u_2 in T_{13} and $\tilde{v}_2$ is the unique level-10 splitting node in the barcode graph. The paths $(\tilde{v}_1 \rightsquigarrow u_1)_{[1,7]}$ and $(\tilde{v}_2 \rightsquigarrow u_2)_{[10,13]}$ are marked with dashes.

▶ **Proposition 13.** *Each interval produced by Algorithm 1 admits a set of 0-representatives.*

Proof. Suppose that an interval is produced by the merge event at iteration i. We have the following situations:

- If the nodes u_1, u_2 in this event (see Algorithm 1) are in the same tree in T_i, let v be the highest common ancestor of u_1, u_2 and note that v is a splitting node at level j. Also note that u_1, u_2 are actually leaves in T_i and hence can also be considered as level-i

nodes in $\mathbb{G}_B(\mathcal{F})$. Let $\tilde{v}$ be the unique level-j splitting node in $\mathbb{G}_B(\mathcal{F})$. By invariant 3 of Proposition 11 along with Proposition 10, there are two sets of partial 0-representatives $\{\alpha_k \mid k \in [j,i]\}, \{\beta_k \mid k \in [j,i]\}$ for $[j,i]$ with $\alpha_j = \tilde{v}$, $\alpha_i = u_1$, $\beta_j = \tilde{v}$, and $\beta_i = u_2$. We claim that $\{\alpha_k + \beta_k \mid k \in [j+1,i]\}$ is a set of 0-representatives for the interval $[j+1,i]$. To prove this, we first note the following obvious facts: (i) $\{\alpha_k + \beta_k \mid k \in [j+1,i]\}$ is a set of partial 0-representatives; (ii) $\alpha_{j+1} + \beta_{j+1} \in \mathsf{ker}(\varphi_j^0)$; (iii) $\alpha_i + \beta_i$ is the non-zero element in $\mathsf{ker}(\varphi_i^0)$. So we only need to show that $\alpha_{j+1} + \beta_{j+1} \neq 0$. Let v_1, v_2 be the two level-$(j+1)$ nodes in $\mathbb{G}_B(\mathcal{F})$ connecting to $\tilde{v}$. Then, α_{j+1} equals v_1 or v_2 and the same for β_{j+1}. To see this, we first show that α_{j+1} can only contain v_1, v_2. For contradiction, suppose instead that α_{j+1} contains a level-$(j+1)$ node x with $x \neq v_1$, $x \neq v_2$. Let $(\tilde{v} \rightsquigarrow u_1)_{[j,i]}$ be the simple path that induces $\{\alpha_k \mid k \in [j,i]\}$ as in Proposition 10 and its proof. Then, x is on the path $(\tilde{v} \rightsquigarrow u_1)_{[j,i]}$ and the two adjacent nodes of x on $(\tilde{v} \rightsquigarrow u_1)_{[j,i]}$ are at level j and $j+2$, in which we let y be the one at level j. Note that $y \neq \tilde{v}$ because x is not equal to v_1 or v_2. Since $(\tilde{v} \rightsquigarrow u_1)_{[j,i]}$ is within level j and i, y must be adjacent to another level-$(j+1)$ node distinct from x on $(\tilde{v} \rightsquigarrow u_1)_{[j,i]}$. Now we have that y is a level-j splitting node with $y \neq \tilde{v}$, contradicting the fact that $\mathbb{G}_B(\mathcal{F})$ has only one level-j splitting node. The fact that α_{j+1} contains v_1 or v_2 but not both can be similarly verified. To see that $\alpha_{j+1} + \beta_{j+1} \neq 0$, suppose instead that $\alpha_{j+1} + \beta_{j+1} = 0$, i.e., $\alpha_{j+1} = \beta_{j+1}$, and without loss of generality they both equal v_1. Note that we can consider T_i as derived by contracting nodes of $\mathbb{G}_B(\mathcal{F}^i)$ at the same level[2]. The fact that $\alpha_{j+1} = \beta_{j+1} = v_1$ implies that u_1, u_2 are descendants of the same child of v in T_i, contradicting the fact that v is the highest common ancestor of u_1, u_2. So we have that $\alpha_{j+1} + \beta_{j+1} \neq 0$.

- If u_1, u_2 are in different trees in T_i, then without loss of generality let u_1 be the one whose root v_1 is at the higher level (i.e., level j). As the root of u_1, the node v_1 must be an entering node, and the connected component of $\mathbb{G}_B(\mathcal{F}^i)$ containing u_1 must have a single level-j node $\tilde{v}_1$. Then, by invariant 2 of Proposition 11 along with Proposition 10, there are two sets of partial 0-representatives $\{\alpha_k \mid k \in [j,i]\}, \{\beta_k \mid k \in [j,i]\}$ for $[j,i]$ with $\alpha_j = \tilde{v}_1$, $\alpha_i = u_1$, $\beta_j = \tilde{v}_2$, and $\beta_i = u_2$, where $\tilde{v}_2$ is a level-j node. We claim that $\{\alpha_k + \beta_k \mid k \in [j,i]\}$ is a set of 0-representatives for the interval $[j,i]$ and the verification is similar to the previous case where u_1 and u_2 are in the same tree.

For intervals produced by the departure events and at the end of the algorithm, the existence of 0-representatives can be similarly argued. ◀

▶ **Theorem 14.** *Algorithm 1 computes the 0-th zigzag barcode for a given zigzag filtration.*

Proof. First, we have the following facts: every level-j entering node in $\mathbb{G}_B(\mathcal{F})$ introduces a $j \in \mathsf{P}(\mathsf{H}_0(\mathcal{F}))$ and uniquely corresponds to a level-j root in T_i for some i; every level-j splitting node in $\mathbb{G}_B(\mathcal{F})$ introduces a $j+1 \in \mathsf{P}(\mathsf{H}_0(\mathcal{F}))$ and uniquely corresponds to a level-j splitting node in T_i for some i. Whenever an interval $[j,i]$ is produced in Algorithm 1, $i \in \mathsf{N}(\mathsf{H}_0(\mathcal{F}))$ and the entering or splitting node in T_i introducing j as a positive index either becomes a *regular* node (i.e., connecting to a single node on both adjacent levels) or is deleted in T_{i+1}. This means that j is never the start of another interval produced. At the end of Algorithm 1, the number of intervals produced which end with m also matches the rank of $\mathsf{H}_0(G_m)$. Therefore, intervals produced by the algorithm induce a bijection $\pi : \mathsf{P}(\mathsf{H}_0(\mathcal{F})) \to \mathsf{N}(\mathsf{H}_0(\mathcal{F}))$. By Proposition 9 and 13, our conclusion follows. ◀

[2] We should further note that this contraction is not done on the entire $\mathbb{G}_B(\mathcal{F}^i)$ but rather on connected components of $\mathbb{G}_B(\mathcal{F}^i)$ containing leaves.

4 One-dimensional zigzag persistence

In this section, we present an efficient algorithm for 1-st zigzag persistence on graphs. We assume that the input is a graph G with a zigzag filtration $\mathcal{F} : \varnothing = G_0 \xleftrightarrow{\sigma_0} G_1 \xleftrightarrow{\sigma_1} \cdots \xleftrightarrow{\sigma_{m-1}} G_m$ of G. We first describe the algorithm without giving the full implementation details. The key to the algorithm is a pairing principle for the positive and negative indices. Then, in Section 4.1, we make several observations which reduce the index pairing to finding the *max edge-weight* of a path in a minimum spanning forest, leading to an efficient implementation.

We notice that the following are true for every inclusion $G_i \xleftrightarrow{\sigma_i} G_{i+1}$ of $\mathcal{F}$ (recall that φ_i^1 denotes the corresponding linear map in the induced module $\mathsf{H}_1(\mathcal{F})$):

- If σ_i is an edge being added and vertices of σ_i are connected in G_i, then φ_i^1 is an injection with non-trivial cokernel, which provides $i+1 \in \mathsf{P}(\mathsf{H}_1(\mathcal{F}))$.
- If σ_i is an edge being deleted and vertices of σ_i are connected in G_{i+1}, then φ_i^1 is an injection with non-trivial cokernel, which provides $i \in \mathsf{N}(\mathsf{H}_1(\mathcal{F}))$.
- In all the other cases, φ_i^1 is an isomorphism and $i \notin \mathsf{N}(\mathsf{H}_1(\mathcal{F}))$, $i+1 \notin \mathsf{P}(\mathsf{H}_1(\mathcal{F}))$.

As can be seen from Section 3, computing $\mathsf{Pers}(\mathsf{H}_1(\mathcal{F}))$ boils down to finding a pairing of indices of $\mathsf{P}(\mathsf{H}_1(\mathcal{F}))$ and $\mathsf{N}(\mathsf{H}_1(\mathcal{F}))$. Our algorithm adopts this structure, where $\mathcal{U}_i$ denotes the set of unpaired positive indices at the *beginning* of each iteration i:

Algorithm 2 Algorithm for 1-st zigzag persistence on graphs.

$\mathcal{U}_0 := \varnothing$
for $i := 0, \ldots, m-1$:
  if $G_i \xrightarrow{\sigma_i} G_{i+1}$ provides $i+1 \in \mathsf{P}(\mathsf{H}_1(\mathcal{F}))$:
    $\mathcal{U}_{i+1} := \mathcal{U}_i \cup \{i+1\}$
  else if $G_i \xleftarrow{\sigma_i} G_{i+1}$ provides $i \in \mathsf{N}(\mathsf{H}_1(\mathcal{F}))$:
    pair i with a $j_* \in \mathcal{U}_i$ based on the Pairing Principle below
    output an interval $[j_*, i]$ for $\mathsf{Pers}(\mathsf{H}_1(\mathcal{F}))$
    $\mathcal{U}_{i+1} := \mathcal{U}_i \setminus \{j_*\}$
  else:
    $\mathcal{U}_{i+1} := \mathcal{U}_i$
for each $j \in \mathcal{U}_m$:
  output an interval $[j, m]$ for $\mathsf{Pers}(\mathsf{H}_1(\mathcal{F}))$

Note that in Algorithm 2, whenever a positive or negative index is produced, σ_i must be an edge. One key piece missing from the algorithm is how we choose a positive index to pair with a negative index:

▶ **Pairing Principle for Algorithm 2.** *In each iteration i where $G_i \xleftarrow{\sigma_i} G_{i+1}$ provides $i \in \mathsf{N}(\mathsf{H}_1(\mathcal{F}))$, let J_i consist of every $j \in \mathcal{U}_i$ such that there exists a 1-cycle z containing both σ_{j-1} and σ_i with $z \subseteq G_k$ for every $k \in [j, i]$. Then, $J_i \neq \varnothing$ and Algorithm 2 pairs i with the* smallest *index j_* in J_i.*

▶ Remark 15. See Proposition 17 for a proof of $J_i \neq \varnothing$ claimed above.

▶ Remark 16. Algorithms for non-zigzag persistence [10, 20] always pair a negative index with the *largest* (i.e., youngest) positive index satisfying a certain condition, while Algorithm 2 pairs with the smallest one. This is due to the difference of zigzag and non-zigzag persistence and our particular condition that 1-cycles can never become boundaries in graphs. See [4, 15] for the pairing when assuming general zigzag filtrations.

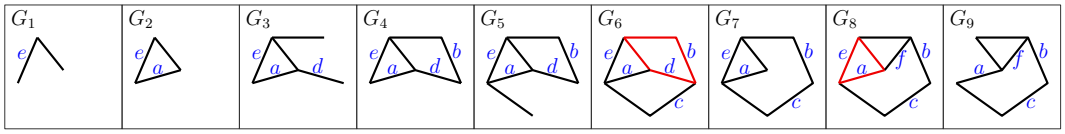


Figure 5 A zigzag filtration with 1-st barcode $\{[4,6],[2,8],[6,9],[8,9]\}$. For brevity, the addition of vertices and some edges are skipped.

Figure 5 gives an example of the pairing of the indices and their corresponding edges. In the figure, when edge d is deleted from G_6, there are three unpaired positive edges a, b, and c, in which b and c admit 1-cycles as required by the Pairing Principle. As the earlier edge, b is paired with d and an interval $[4,6]$ is produced. The red cycle in G_6 indicates the 1-cycle containing b and d which exists in all the intermediate graphs. Similar situations happen when e is paired with a in G_8, producing the interval $[2,8]$.

For the correctness of Algorithm 2, we first provide Proposition 17 which justifies the Pairing Principle and is a major step leading toward our conclusion (Theorem 18):

▶ **Proposition 17.** *At the beginning of each iteration i in Algorithm 2, for every $j \in \mathcal{U}_i$, there exists a 1-cycle z_j^i containing σ_{j-1} with $z_j^i \subseteq G_k$ for every $k \in [j,i]$. Furthermore, the set $\{z_j^i \mid j \in \mathcal{U}_i\}$ forms a basis of $\mathsf{Z}_1(G_i)$. If the iteration i produces a negative index i, then the above statements imply that there is at least one z_j^i containing σ_i. This z_j^i satisfies the condition that $z_j^i \subseteq G_k$ for every $k \in [j,i]$, $\sigma_{j-1} \in z_j^i$, and $\sigma_i \in z_j^i$, which implies that $J_i \neq \varnothing$ where J_i is as defined in the Pairing Principle.*

▶ **Theorem 18.** *Algorithm 2 computes the 1-st zigzag barcode for a given zigzag filtration on graphs.*

Proof. The claim follows directly from Proposition 9. For each interval $[j_*,i]$ produced from the pairing in Algorithm 2, by the Pairing Principle, there exists a 1-cycle z containing both σ_{j_*-1} and σ_i with $z \subseteq G_k$ for every $k \in [j_*,i]$. The cycle z induces a set of 1-representatives for $[j_*,i]$. For each interval produced at the end, Proposition 17 implies that such an interval admits 1-representatives. ◀

4.1 Efficient implementation

For every i and every $j \leq i$, define Γ_j^i as the graph derived from G_j by deleting every edge σ_k s.t. $j \leq k < i$ and $G_k \xleftarrow{\sigma_k} G_{k+1}$ is backward. For convenience, we also assume that Γ_j^i contains all the vertices of G. We can simplify the Pairing Principle as suggested by the following proposition:

▶ **Proposition 19.** *In each iteration i of Algorithm 2 where $G_i \xleftarrow{\sigma_i} G_{i+1}$ provides $i \in \mathsf{N}(\mathsf{H}_1(\mathcal{F}))$, the set J_i in the Pairing Principle can be alternatively defined as consisting of every $j \in \mathcal{U}_i$ s.t. $\sigma_i \in \Gamma_j^i$ and the vertices of σ_i are connected in Γ_j^{i+1} ($\sigma_i \notin \Gamma_j^{i+1}$ by definition).*

We now turn graphs in $\mathcal{F}$ into weighted ones in the following way: initially, $G_0 = \varnothing$; then, whenever an edge σ_i is added from G_i to G_{i+1}, the weight $w(\sigma_i)$ is set to i. For a path in a weighted graph, let the *max edge-weight* of the path be the maximum weight of its edges. Then, as can be seen from the full version [7] of the paper, the pairing boils down to computing the max edge-weight of the path connecting u, v in the minimum spanning forest (MSF) of G_{i+1}. For this, we utilize the *dynamic-MSF* data structure proposed by Holm et al. [12]. Assuming that n is the number of vertices and edges of G, the dynamic-MSF data structure supports the following operations:

- Return the identifier of a vertex's connected component in $O(\log n)$ time, which can be used to determine whether two vertices are connected.
- Return the max edge-weight of the path connecting any two vertices in the MSF in $O(\log n)$ time.
- Insert or delete an edge from the current graph (maintained by the data structure) and possibly update the MSF in $O(\log^4 n)$ amortized time.

We now present the full details of Algorithm 2:

Algorithm Algorithm 2: details.

Maintain a dynamic-MSF data structure $\mathbb{F}$, which consists of all vertices of G and no edges initially. Also, set $\mathcal{U}_0 = \varnothing$. Then, for each $i = 0, \ldots, m-1$, if σ_i is a vertex, do nothing; otherwise, do the following:

Case $G_i \xrightarrow{\sigma_i} G_{i+1}$: Check whether vertices of σ_i are connected in G_i by querying $\mathbb{F}$, and then add σ_i to $\mathbb{F}$. If vertices of σ_i are connected in G_i, then set $\mathcal{U}_{i+1} = \mathcal{U}_i \cup \{i+1\}$; otherwise, set $\mathcal{U}_{i+1} = \mathcal{U}_i$.

Case $G_i \xleftarrow{\sigma_i} G_{i+1}$: Delete σ_i from $\mathbb{F}$. If the vertices u, v of σ_i are found to be not connected in G_{i+1} by querying $\mathbb{F}$, then set $\mathcal{U}_{i+1} = \mathcal{U}_i$; otherwise, do the following:

- Find the max edge-weight w_* of the path connecting u, v in the MSF of G_{i+1} by querying $\mathbb{F}$.
- Find the smallest index j_* of $\mathcal{U}_i$ greater than $\max\{w_*, w(\sigma_i)\}$. (Note that we can store $\mathcal{U}_i$ as a red-black tree [5], so that finding j_* takes $O(\log n)$ time.)
- Output an interval $[j_*, i]$ and set $\mathcal{U}_{i+1} = \mathcal{U}_i \setminus \{j_*\}$.

At the end, for each $j \in \mathcal{U}_m$, output an interval $[j, m]$.

Now we can see that Algorithm 2 has time complexity $O(m \log^4 n)$, where each iteration is dominated by the update of $\mathbb{F}$.

5 Codimension-one zigzag persistence of embedded complexes

In this section, we present an efficient algorithm for computing the $(p-1)$-th barcode given a zigzag filtration of an $\mathbb{R}^p$-embedded complex, by extending our algorithm for 0-dimension with the help of Alexander duality [17].

Throughout this section, $p \geq 2$, K is a simplicial complex embedded in $\mathbb{R}^p$, and $\mathcal{F} : \varnothing = K_0 \leftrightarrow \cdots \leftrightarrow K_m$ is a zigzag filtration of K. We call connected components of $\mathbb{R}^p \setminus |K|$ as *voids* of K or *K-voids*, to emphasize that only voids of K are considered in this section. The *dual graph* G of K has the vertices corresponding to the voids as well as the p-simplices of K, and has the edges corresponding to the $(p-1)$-simplices of K. The *dual filtration* $\mathcal{E} : G = G_0 \leftrightarrow G_1 \leftrightarrow \cdots \leftrightarrow G_m$ of $\mathcal{F}$ consists of subgraphs G_i of G such that: (i) all vertices of G dual to a K-void are in G_i; (ii) a vertex of G dual to a p-simplex is in G_i iff the dual p-simplex is *not* in K_i; (iii) an edge of G is in G_i iff its dual $(p-1)$-simplex is *not* in K_i.

One could verify that each G_i is a well-defined subgraph of G. We note the following: (i) inclusion directions in $\mathcal{E}$ are reversed; (ii) $\mathcal{E}$ is not exactly a zigzag filtration (because an arrow may introduce no changes) but can be easily made into one. Figure 6 gives an example in $\mathbb{R}^2$, in which we observe the following: whenever a $(p-1)$-cycle (i.e., 1-cycle) is formed in the primal filtration, a connected component in the dual filtration splits; whenever

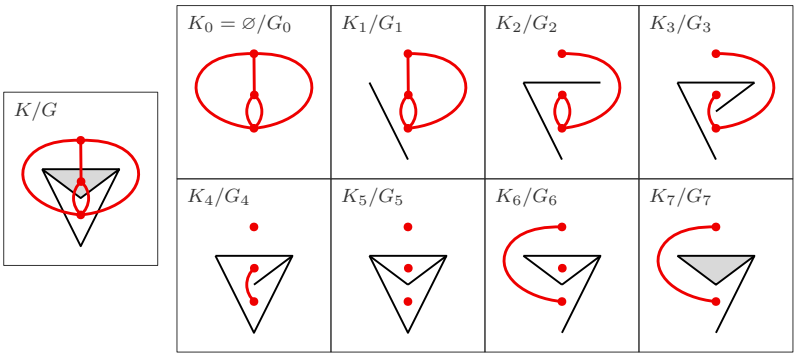


Figure 6 A zigzag filtration of an $\mathbb{R}^2$-embedded complex K and its dual filtration, where the dual graphs are colored red. For brevity, changes of vertices in the primal filtration are ignored.

a $(p-1)$-cycle is killed in the primal filtration, a connected component in the dual filtration vanishes. Intuitively, G_i encodes the connectivity of $\mathbb{R}^p \setminus |K_i|$, and so by Alexander duality, we have the following proposition:

▶ **Proposition 20.** $\mathsf{Pers}(\mathsf{H}_{p-1}(\mathcal{F})) = \mathsf{Pers}(\tilde{\mathsf{H}}_0(\mathcal{E}))$.

Proposition 21 indicates a way to compute $\mathsf{Pers}(\tilde{\mathsf{H}}_0(\mathcal{E}))$:

▶ **Proposition 21.** $\mathsf{Pers}(\tilde{\mathsf{H}}_0(\mathcal{E})) = \mathsf{Pers}(\mathsf{H}_0(\mathcal{E})) \setminus \{[0, m]\}$.

The above two propositions suggest a naive algorithm for computing $\mathsf{Pers}(\mathsf{H}_{p-1}(\mathcal{F}))$ using Algorithm 1. However, building the dual graph G requires reconstructing the void boundaries of K, which is done by a "walking" algorithm obtaining a set of $(p-1)$-cycles and then by a nesting test of these $(p-1)$-cycles [8, Section 4.1]. The running time of this process is $\Omega(n^2)$ where n is the size of K. To achieve the claimed complexity, we first define the following [8]:

▶ **Definition 22.** *In a simplicial complex X, two q-simplices σ, σ' are* ***q-connected*** *if there is a sequence $\sigma_0, \ldots, \sigma_\ell$ of q-simplices of X such that $\sigma_0 = \sigma$, $\sigma_\ell = \sigma'$, and every σ_i, σ_{i+1} share a $(q-1)$-face. A maximal set of q-connected q-simplices of X is called a* ***q-connected component*** *of X, and X is* ***q-connected*** *if it has only one q-connected component.*

Based on the fact that void boundaries can be reconstructed without the nesting test for $(p-1)$-connected complexes in $\mathbb{R}^p$ [8], we restrict $\mathcal{F}$ to several $(p-1)$-connected subcomplexes of K and then take the union of the $(p-1)$-th barcodes of these restricted filtrations:

Algorithm 3 Algorithm for $(p-1)$-th zigzag persistence on $\mathbb{R}^p$-embedded complexes.

1. Compute the $(p-1)$-connected components $D^1, \ldots, D^r$ of K.
2. For each $\ell = 1, \ldots, r$, let

$$C^\ell = \mathrm{cls}(D^\ell) \cup \{\tau \in K \mid \tau \text{ is a } p\text{-simplex whose } (p-1)\text{-faces are in } D^\ell\}$$

and let $\mathcal{X}^\ell$ be a filtration of C^ℓ defined as

$$\mathcal{X}^\ell : K_0 \cap C^\ell \leftrightarrow K_1 \cap C^\ell \leftrightarrow \cdots \leftrightarrow K_m \cap C^\ell$$

Then, compute $\mathsf{Pers}(\mathsf{H}_{p-1}(\mathcal{X}^\ell))$ for each ℓ.
3. Return $\bigcup_{\ell=1}^{r} \mathsf{Pers}(\mathsf{H}_{p-1}(\mathcal{X}^\ell))$ as the $(p-1)$-th barcode of $\mathcal{F}$.

In Algorithm 3, $\mathrm{cls}(D^\ell)$ denotes the closure of D^ℓ, i.e., the complex consisting of all faces of simplices in D^ℓ. For each ℓ, let n_ℓ be the number of simplices in C^ℓ; then, the dual graph of C^ℓ can be constructed in $O(n_\ell \log n_\ell)$ time because C^ℓ is $(p-1)$-connected [8]. Using Algorithm 1 to compute $\mathsf{Pers}(\mathsf{H}_{p-1}(\mathcal{X}^\ell))$ as suggested by the duality, the running time of Algorithm 3 is $O(m \log^2 n + m \log m + n \log n)$, where m is the length of $\mathcal{F}$ and n is the size of K.

The correctness of Algorithm 3 can be seen from the following proposition:

▶ **Proposition 23.** *The modules* $\mathsf{H}_{p-1}(\mathcal{F})$ *and* $\bigoplus_{\ell=1}^{r} \mathsf{H}_{p-1}(\mathcal{X}^\ell)$ *are isomorphic which implies that* $\mathsf{Pers}(\mathsf{H}_{p-1}(\mathcal{F})) = \bigcup_{\ell=1}^{r} \mathsf{Pers}(\mathsf{H}_{p-1}(\mathcal{X}^\ell))$.

References

1 Pankaj K. Agarwal, Herbert Edelsbrunner, John Harer, and Yusu Wang. Extreme elevation on a 2-manifold. *Discrete & Computational Geometry*, 36(4):553–572, 2006.

2 Josh Alman and Virginia Vassilevska Williams. A refined laser method and faster matrix multiplication. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 522–539. SIAM, 2021.

3 Gunnar Carlsson and Vin de Silva. Zigzag persistence. *Foundations of Computational Mathematics*, 10(4):367–405, 2010.

4 Gunnar Carlsson, Vin de Silva, and Dmitriy Morozov. Zigzag persistent homology and real-valued functions. In *Proceedings of the Twenty-Fifth Annual Symposium on Computational Geometry*, pages 247–256, 2009.

5 Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, 3rd Edition*. MIT Press, 2009. URL: `http://mitpress.mit.edu/books/introduction-algorithms`.

6 Tamal K. Dey. Computing height persistence and homology generators in $\mathbb{R}^3$ efficiently. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2649–2662. SIAM, 2019.

7 Tamal K. Dey and Tao Hou. Computing zigzag persistence on graphs in near-linear time. *arXiv preprint*, 2021. `arXiv:2103.07353`.

8 Tamal K. Dey, Tao Hou, and Sayan Mandal. Computing minimal persistent cycles: Polynomial and hard cases. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2587–2606. SIAM, 2020.

9 Herbert Edelsbrunner and Michael Kerber. Alexander duality for functions: the persistent behavior of land and water and shore. In *Proceedings of the Twenty-Eighth Annual Symposium on Computational Geometry*, pages 249–258, 2012.

10 Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological persistence and simplification. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 454–463. IEEE, 2000.

11 Loukas Georgiadis, Haim Kaplan, Nira Shafrir, Robert E. Tarjan, and Renato F. Werneck. Data structures for mergeable trees. *ACM Transactions on Algorithms (TALG)*, 7(2):1–30, 2011.

12 Jacob Holm, Kristian De Lichtenberg, and Mikkel Thorup. Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity. *Journal of the ACM (JACM)*, 48(4):723–760, 2001.

13 Petter Holme and Jari Saramäki. Temporal networks. *Physics Reports*, 519(3):97–125, 2012.

14 Woojin Kim and Facundo Memoli. Stable signatures for dynamic graphs and dynamic metric spaces via zigzag persistence. *arXiv preprint*, 2017. `arXiv:1712.04064`.

15 Clément Maria and Steve Y. Oudot. Zigzag persistence via reflections and transpositions. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 181–199. SIAM, 2014.

16 Nikola Milosavljević, Dmitriy Morozov, and Primoz Skraba. Zigzag persistent homology in matrix multiplication time. In *Proceedings of the Twenty-Seventh Annual Symposium on Computational Geometry*, pages 216–225, 2011.

17 James R. Munkres. *Elements of Algebraic Topology*. CRC Press, 2018.

18 Benjamin Schweinhart. *Statistical Topology of Embedded Graphs*. PhD thesis, Princeton University Press, 2015.

19 Joakim Skarding, Bogdan Gabrys, and Katarzyna Musial. Foundations and modelling of dynamic networks using dynamic graph neural networks: A survey. *arXiv preprint*, 2020. `arXiv:2005.07496`.

20 Afra Zomorodian and Gunnar Carlsson. Computing persistent homology. *Discrete & Computational Geometry*, 33(2):249–274, 2005.

Minimal Delaunay Triangulations of Hyperbolic Surfaces

Matthijs Ebbens
Bernoulli Institute for Mathematics, Computer Science and Artificial Intelligence, University of Groningen, The Netherlands

Hugo Parlier
Mathematics Research Unit, University of Luxembourg, Luxembourg

Gert Vegter
Bernoulli Institute for Mathematics, Computer Science and Artificial Intelligence, University of Groningen, The Netherlands

Abstract

Motivated by recent work on Delaunay triangulations of hyperbolic surfaces, we consider the minimal number of vertices of such triangulations. First, we show that every hyperbolic surface of genus g has a simplicial Delaunay triangulation with $O(g)$ vertices, where edges are given by distance paths. Then, we construct a class of hyperbolic surfaces for which the order of this bound is optimal. Finally, to give a general lower bound, we show that the $\Omega(\sqrt{g})$ lower bound for the number of vertices of a simplicial triangulation of a topological surface of genus g is tight for hyperbolic surfaces as well.

2012 ACM Subject Classification Theory of computation → Computational geometry; Mathematics of computing → Geometric topology; Mathematics of computing → Graphs and surfaces

Keywords and phrases Delaunay triangulations, hyperbolic surfaces, metric graph embeddings, moduli spaces

Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.31

Related Version *Full Version*: `https://arxiv.org/abs/2011.09847`

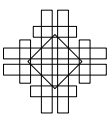
Funding *Hugo Parlier*: Research partially supported by ANR/FNR project SoS, INTER/ANR/16/11554412/SoS, ANR-17-CE40-0033.

Acknowledgements The authors warmly thank Monique Teillaud and Vincent Despré for fruitful discussions. We would also like to thank the referees for their helpful comments.

1 Introduction

The classical topic of Delaunay triangulations has recently been studied in the context of hyperbolic surfaces. Bowyer's incremental algorithm for computing simplicial Delaunay triangulations in the Euclidean plane [5] has been generalized to orientable hyperbolic surfaces and implemented for some specific cases [4, 11]. Moreover, it has been shown that the flip graph of geometric (but not necessarily simplicial) Delaunay triangulations on a hyperbolic surface is connected [7].

In this work, we consider the minimal number of vertices of a simplicial Delaunay triangulation of a closed hyperbolic surface of genus g. Motivated by the interest in embeddings where edges are shortest paths between their endpoints [8, 10], which have applications in for example the field of graph drawing [17], we restrict ourselves to *distance* Delaunay triangulations, where edges are distance paths.



Our main result is the upper bound on the number of vertices with sharp order of growth:

▶ **Theorem 1.** *An orientable closed hyperbolic surface of genus $g \geq 2$ has a distance Delaunay triangulation with at most $O(g)$ vertices. There exists a family of surfaces, X_g, $g \geq 2$, such that the number of vertices of any distance Delaunay triangulation of them grows like $\Omega(g)$.*

The above result is a compilation of Theorems 4 and 18 where explicit upper and lower bounds are given.

Another reason to study triangulations whose edges are distance paths, comes from the study of moduli spaces $\mathcal{M}_g$, which we can think of as a space of all hyperbolic surfaces of genus $g \geq 2$ up to isometry. These spaces admit natural coordinates associated to pants decompositions (the so-called Fenchel-Nielsen coordinates, see Section 2 for details). It is a classical theorem of Bers [2] that any surface admits a short pants decomposition, meaning that the length of each of its simple closed geodesics is bounded by a function that only depends on the topology of the surface (but not its geometry). As these curves provide a local description of the surface, one might hope that they are also geodesically convex, meaning that the shortest distance path between any two points of a given curve is contained in the curve. It is perhaps surprising that most surfaces admit no short pants decompositions with geodesically convex curves. Indeed it is known that *any* pants decomposition of a random surface (chosen with respect to a natural probability measure on $\mathcal{M}_g$) has at least one curve of length on the order of $g^{\frac{1}{6}-\varepsilon}$ as g grows (for any fixed $\varepsilon > 0$) [9]. And it is a theorem of Mirzakhani that these same random surfaces are also of *diameter* on the order of $\log(g)$ [13]. Hence the longest curve of any pants decomposition of a random surface is not convex.

The lengths of edges in a given triangulation are another parameter set for $\mathcal{M}_g$. By the theorem above, such a parameter set can be chosen with a reasonable number of vertices such that the edges are all convex. Using the moduli space point of view, one has a function $\omega : \mathcal{M}_g \to \mathbb{N}$ which associates to a surface the minimal number of vertices of any of its distance Delaunay triangulations. The above result implies that

$$\limsup_{g\to\infty} \max_{X\in\mathcal{M}_g} \frac{\omega(X)}{g}$$

is finite and strictly positive, but for instance we do not know whether the actual limit exists.

The examples we exhibit are geometrically quite simple, as they are made by gluing hyperbolic pants, with bounded cuff lengths, in something that resembles a line as the genus grows. One might wonder whether all surfaces have this property, but we show this is not the case by exploring the quantity $\min_{X\in\mathcal{M}_g} \omega(X)$. This quantity has a precise lower bound on the order of $\Theta(\sqrt{g})$ because we ask that our triangulations be simplicial [12]. We show how to use the celebrated Ringel-Youngs construction [15] to construct a family of hyperbolic surfaces that attain this bound for infinitely many genera (Theorem 26), showing that one cannot hope for better than the simplicial lower bound in general.

This paper is structured as follows. In Section 2, we introduce our notation and give some preliminaries on hyperbolic surface theory and triangulations. In Section 3, we prove our linear upper bound for the number of vertices of a minimal distance Delaunay triangulation. In Section 4, we construct classes of hyperbolic surfaces attaining the order of this linear upper bound. Finally, in Section 5, we construct a family of hyperbolic surfaces attaining the general $\Theta(\sqrt{g})$ lower bound.

2 Preliminaries

We will start by recalling some hyperbolic geometry. There are several models for the hyperbolic plane [1]. In the Poincaré disk model, the hyperbolic plane is represented by the unit disk $\mathbb{D}$ in the complex plane equipped with a specific Riemannian metric of constant Gaussian curvature -1. With respect to this metric, hyperbolic lines, i.e., geodesics are given by diameters of $\mathbb{D}$ or circle segments intersecting $\partial\mathbb{D}$ orthogonally. A hyperbolic circle is a Euclidean circle contained in $\mathbb{D}$. However, in general the centre and radius of a hyperbolic circle are different from the Euclidean centre and radius.

A *hyperbolic surface* is a 2-dimensional Riemannian manifold that is locally isometric to an open subset of the hyperbolic plane [6, 16], thus of constant curvature -1. Our surfaces are assumed throughout to be *closed* and *orientable*, and because they are hyperbolic, via Gauss-Bonnet, their genus g satisfies $g \geq 2$ and their area is $4\pi(g-1)$. Note that we will frequently be interested in subsurfaces of a closed surface which we think of as compact surfaces with boundary consisting of a collection of simple closed geodesics. The signature of such a subsurface is (g', k) where g' is its genus and k is the number of boundary geodesics.

Via the uniformization theorem, any hyperbolic surface X can be written as a quotient space $X = \mathbb{D}/\Gamma$ of the hyperbolic plane under the action of a Fuchsian group Γ (a discrete subgroup of the group of orientation-preserving isometries of $\mathbb{D}$). The hyperbolic plane $\mathbb{D}$ is the *universal cover* of X and is equipped with a projection $\pi : \mathbb{D} \to \mathbb{D}/\Gamma$.

In the free homotopy class of any non-contractible closed curve on a hyperbolic surface lies a unique closed geodesic. If the curve is simple, then the corresponding geodesic is simple, and hence it is a straightforward topological exercise to decompose a hyperbolic surface into $2g-2$ pairs of pants by cutting along $3g-3$ disjoint simple closed geodesics (Figure 1). A *pair of pants* is a surface homeomorphic to a three times punctured sphere but we generally think of its closure, and thus of a hyperbolic pair of pants as being a surface of genus 0 with three simple closed geodesics as boundary, i.e., a surface of signature $(0, 3)$.

It is a short but useful exercise in hyperbolic trigonometry to show that a hyperbolic pair of pants is determined by its three boundary lengths. Hence, the lengths of the $3g-3$ geodesics determine the geometry of each of the $2g-2$ pairs of pants, but to determine X, one needs to add twist parameters that control how the pants are pasted together. How one computes the twist coordinate is at least partially a matter of taste, and although we will not make much use of it, for completeness we follow [6], where the twist is the signed distance between marked points on the boundary curves.

The length and twist parameters determine X and are called Fenchel-Nielsen coordinates. These parameters can be chosen freely in the set $(\mathbb{R}^{>0})^{3g-3} \times \mathbb{R}^{3g-3}$. What they determine is more than just an isometry class of a surface: they determine a *marked* hyperbolic surface, homeomorphic to a base topological surface Σ. As the lengths and twists change, the marked surface changes, and the Fenchel-Nielsen coordinates provide a parameter set for the space of marked hyperbolic surfaces of genus g, called *Teichmüller space* $\mathcal{T}_g$. The underlying *moduli space* $\mathcal{M}_g$ can be thought of as the space of hyperbolic surfaces up to isometry, obtained from $\mathcal{T}_g$ by "forgetting" the marking.

Throughout the paper, lengths of closed geodesics will play an important role. As mentioned above, in the free homotopy class of a non-contractible closed curve lies a unique geodesic representative, and as the metric changes, the length of the geodesic changes, but the free homotopy class does not. Generally we will be dealing with a fixed surface $X \in \mathcal{T}_g$, and the length of a geodesic γ will be denoted by $\ell(\gamma)$. Nonetheless, it is sometimes useful to think of the length of the corresponding homotopy class as a function over $\mathcal{T}_g$ which associates to X the length of the geodesic corresponding to γ.

To a pair of pants decomposition, we associate a 3-regular graph, where each pair of pants is represented by a vertex and two vertices share an edge if the corresponding pairs of pants share a boundary geodesic (see Figure 2). As our parametrization of $\mathcal{T}_g$ depends on a choice of pair of pants decomposition, one can think of the Fenchel-Nielsen coordinates as associating a length and a twist to each edge.

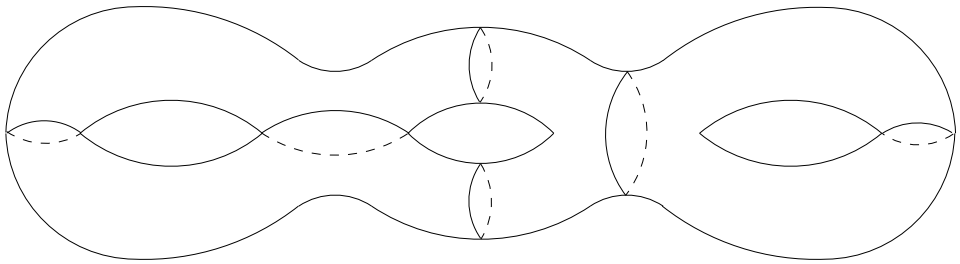


Figure 1 Decomposition of a genus 3 surface into 4 pair of pants using 6 disjoint simple closed geodesics.

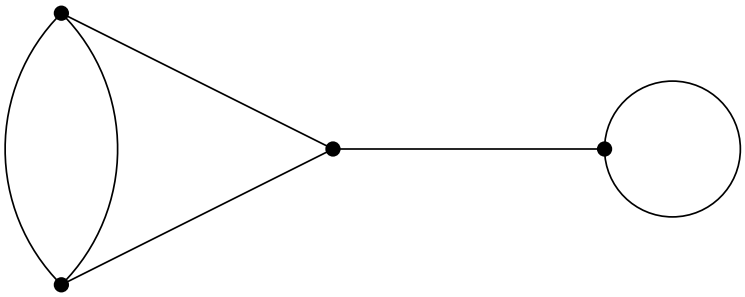


Figure 2 3-regular graph corresponding to the pair of pants decomposition shown in Figure 1.

Around a simple closed geodesic γ, the local geometry of a surface is given by its so-called *collar*. Roughly speaking, for small enough r, the set $C_\gamma(r) = \{x \in X \mid d(x,\gamma) \leq r\}$ is an embedded cylinder. A bound on how large one can take the r to be while retaining the cylinder topology is given by the Collar Lemma:

▶ **Lemma 2** ([6, Theorem 4.1.1]). *Let γ by a simple closed geodesic on a closed hyperbolic surface X. The collar $C_\gamma(w(\gamma))$ of width $w(\gamma)$ given by*

$$w(\gamma) = \operatorname{arcsinh}\left(\frac{1}{\sinh(\frac{1}{2}\ell(\gamma))}\right) \tag{1}$$

is an embedded hyperbolic cylinder isometric to $[-w(\gamma), w(\gamma)] \times \mathbb{S}^1$ with the Riemannian metric $ds^2 = d\rho^2 + \ell^2(\gamma)\cosh^2(\rho)dt^2$ at (ρ, t). Furthermore, if two simple closed geodesics γ and γ' are disjoint, then the collars $C_\gamma(w(\gamma))$ and $C_{\gamma'}(w(\gamma'))$ are disjoint as well.

This paper is about *distance Delaunay triangulations* on closed hyperbolic surfaces.

▶ **Definition 3.** *A* distance Delaunay triangulation *is a triangulation satisfying the following three properties:*

1. *it is a simplicial complex,*
2. *it is a Delaunay triangulation,*
3. *its edges are distance paths.*

The set of all distance Delaunay triangulations of a closed hyperbolic surface X is denoted by $\mathcal{D}(X)$.

We will describe each of these three properties in more detail below.

Simplicial complexes. We will use the standard definition of a simplicial complex. In our case, an embedding of a graph into a surface is a simplicial complex if and only if it does not contain any 1- or 2-cycles. In particular, a geodesic triangulation of a point set in the Euclidean or hyperbolic planes is always a simplicial complex. This is because there are no geodesic monogons or bigons.

Delaunay triangulations. Given a set of vertices in the Euclidean plane, a triangle is called a *Delaunay triangle* if its circumscribed disk does not contain any vertex in its interior. A triangulation of a set of vertices in the Euclidean plane is a *Delaunay triangulation* if all triangles are Delaunay triangles. Using the correspondence between hyperbolic and Euclidean circles, we define Delaunay triangulations in the hyperbolic plane similarly.

Delaunay triangulations on hyperbolic surfaces can be defined by lifting vertices on a hyperbolic surface X to the universal cover $\mathbb{D}$ [3, 7]. More specifically, let $\mathcal{P}$ be a set of vertices on X and let $\pi : \mathbb{D} \to \mathbb{D}/\Gamma$ be the projection of the hyperbolic plane $\mathbb{D}$ to the hyperbolic surface $X = \mathbb{D}/\Gamma$. A triangle (v_1, v_2, v_3) with $v_i \in \mathcal{P}$ is called a *Delaunay triangle* if there exist pre-images $v_i' \in \pi^{-1}(\{v_i\})$ such that the circumscribed disk of the triangle (v_1', v_2', v_3') in the hyperbolic plane does not contain any point of $\pi^{-1}(\mathcal{P})$ in its interior. A triangulation of $\mathcal{P}$ on X is a *Delaunay triangulation* if all triangles are Delaunay triangles.

A Delaunay triangulation of a point set on a hyperbolic surface X is related to a Delaunay triangulation in $\mathbb{D}$ as follows [3]. Given a point set $\mathcal{P}$ on X, we consider a Delaunay triangulation T' of the infinite point set $\pi^{-1}(\mathcal{P})$. Then, we let $T = \pi(T')$. By definition, T is a Delaunay triangulation. Moreover, because every triangulation in $\mathbb{D}$ is a simplicial complex, T' is a simplicial complex. However, T is not necessarily a simplicial complex, because projecting T' to X might introduce 1- or 2-cycles. We will use the correspondence between Delaunay triangulations in $\mathbb{D}$ and in X in Definition 11 and the proof of Theorem 4 and show explicitly that in these cases the result after projecting to X is simplicial.

To make sure that $T = \pi(T')$ is a well-defined triangulation, we will assume without loss of generality that T' is Γ-*invariant*, i.e., the image of any Delaunay triangle in T' under an element of Γ is a Delaunay triangle. Otherwise, it is possible that in so-called *degenerate cases* T contains edges that intersect in a point that is not a vertex [4]. Namely, suppose that T' contains a polygon $P = \{p_1, p_2, \ldots, p_k\}$ consisting of $k \geq 4$ concircular vertices and let T_P be the Delaunay triangulation of P in T'. Because the Delaunay triangulation of a set of at least four concircular vertices is not uniquely defined, assume that there exists $A \in \Gamma$ such that the Delaunay triangulation $T_{A(P)}$ of $A(P)$ in T' is not equal to $A(T_P)$. Because $\pi(P) = \pi(A(P))$, there exists an edge of $\pi(T_{A(P)})$ and an edge of $\pi(A(T_P))$ that intersect in a point that is not a vertex.

Distance paths. Suppose we are given an edge (u, v) in a triangulation of a hyperbolic surface X. Because (u, v) is embedded in X, there exists a geodesic $\gamma : [0, 1] \to X$ with $\gamma(0) = u$ and $\gamma(1) = v$. We say that (u, v) is a *distance path* if $\ell(\gamma) = d(u, v)$, where $d(u, v)$ is the infimum of the lengths of all curves joining u to v.

3 Linear upper bound for the number of vertices of a minimal distance Delaunay triangulation

As our first result, we prove that for every hyperbolic surface there exists a distance Delaunay triangulation with $O(g)$ vertices. Note that the constant 151 is certainly not optimal.

▶ **Theorem 4.** *For every closed hyperbolic surface X of genus g there exists a distance Delaunay triangulation $T \in \mathcal{D}(X)$ with at most $151g$ vertices.*

The idea of the proof is the following. Given a hyperbolic surface X, we construct a vertex set $\mathcal{P}$ on X consisting of at most $151g$ vertices such that the projection T of a Delaunay triangulation of $\pi^{-1}(\mathcal{P})$ in $\mathbb{D}$ to X is a distance Delaunay triangulation of X.

It is known that T is a simplicial complex if $\mathcal{P}$ is sufficiently dense and well-distributed [3]. More precisely, there are no 1- or 2-cycles in T if the diameter of the largest disk in $\mathbb{D}$ not containing any points of $\pi^{-1}(\mathcal{P})$ is less than $\frac{1}{2}\operatorname{sys}(X)$, where $\operatorname{sys}(X)$ is the systole of X, i.e. the length of the shortest homotopically non-trivial closed curve. However, the systole of a hyperbolic surface can be arbitrarily close to zero, which means that we would need an arbitrarily dense set $\mathcal{P}$ to satisfy this condition.

Instead, for a constant $\varepsilon > 0$ we subdivide X into its *ε-thick* part $X^{\varepsilon}_{\text{thick}} = \{x \in X \mid \operatorname{injrad}(x) > \varepsilon\}$ and its *ε-thin* part $X^{\varepsilon}_{\text{thin}} = X \setminus X^{\varepsilon}_{\text{thick}}$, where $\operatorname{injrad}(x)$ is the injectivity radius at x, i.e., the radius of the largest embedded open disk centered at x. Note that the minimum of $\operatorname{injrad}(x)$ over all $x \in X$ is given by $\frac{1}{2}\operatorname{sys}(X)$. We will see in Section 3.1 that, for sufficiently small ε, $X^{\varepsilon}_{\text{thin}}$ is a collection of hyperbolic cylinders (see Figure 3). In these hyperbolic cylinders we want to construct a set of vertices the cardinality of which does not depend on $\operatorname{sys}(X)$. To do this, we put three vertices on the "waist" and each of the two boundary components of the cylinders that are "long and narrow" (for definitions of "waist" and "long and narrow", see Section 3.1). In the cylinders that are not "long and narrow" it suffices to place three vertices on its waist only. Because $\operatorname{injrad}(x) > \varepsilon$ for all $x \in X^{\varepsilon}_{\text{thick}}$, we can construct a sufficiently dense and well-distributed point set in $X^{\varepsilon}_{\text{thick}}$ whose cardinality does not depend on $\operatorname{sys}(X)$ but only on ε. In Section 3.2 we will describe how we combine the vertices placed in the hyperbolic cylinders with the dense and well-distributed set of vertices in $X^{\varepsilon}_{\text{thick}}$. Finally, the proof of Theorem 4 is given in Section 3.3.

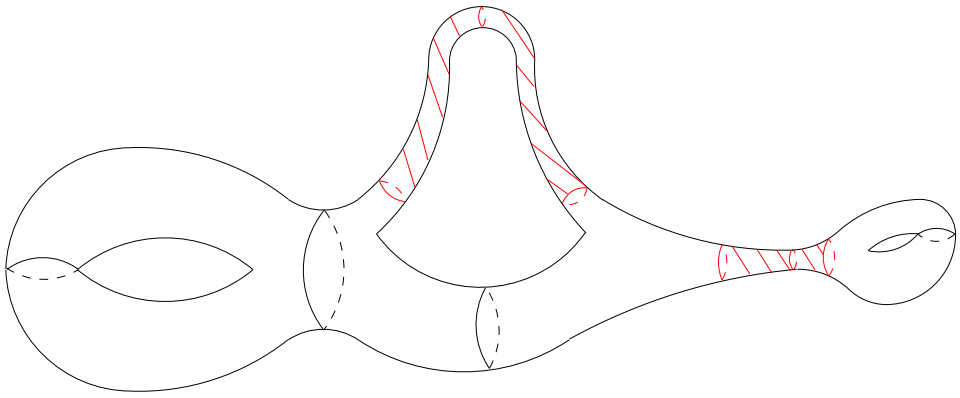


Figure 3 Decomposition of a hyperbolic surface into a thick part consisting of two connected components and two narrow hyperbolic cylinders (in red).

3.1 Distance Delaunay triangulations of hyperbolic cylinders

We now describe our construction of a set of vertices for the ε-thin part $X^{\varepsilon}_{\text{thin}}$ of the hyperbolic surface X. The following lemma describes $X^{\varepsilon}_{\text{thin}}$ in more detail.

▶ **Lemma 5** ([6, Theorem 4.1.6])**.** *If $\varepsilon < \operatorname{arcsinh}(1)$ then $X^{\varepsilon}_{\text{thin}}$ is a collection of at most $3g-3$ pairwise disjoint hyperbolic cylinders.*

Following the description of the geometry of the hyperbolic cylinders in [14], each hyperbolic cylinder C in $X^{\varepsilon}_{\text{thin}}$ consists of points with injectivity radius at most ε and the boundary curves γ^+ and γ^- consist of all points with injectivity radius equal to ε. Every point on the boundary curves is the base point of an embedded geodesic loop of length 2ε (Figure 4), which is completely contained in the hyperbolic cylinder. All points on the boundary curves have the same distance K_C to a closed geodesic γ (called the *waist* of C), where K_C only depends on ε and the length $\ell(\gamma)$ of γ. To see this, fix a point p on γ^+ and consider a distance path ξ from p to γ (Figure 4). Cutting along γ, ξ and the loop of length 2ε with base point p yields a hyperbolic quadrilateral. The common orthogonal of γ and the geodesic loop subdivides this quadrilateral into two congruent quadrilaterals, each with three right angles. Applying a standard result from hyperbolic trigonometry yields $\sinh(\varepsilon) = \sinh(\frac{1}{2}\ell(\gamma))\cosh(\ell(\xi))$ (see, e.g., [6, Formula Glossary 2.3.1(v)]). Because $K_C = \ell(\xi)$, it follows that

$$K_C = \operatorname{arccosh}\left(\frac{\sinh(\varepsilon)}{\sinh(\frac{1}{2}\ell(\gamma))}\right). \tag{2}$$

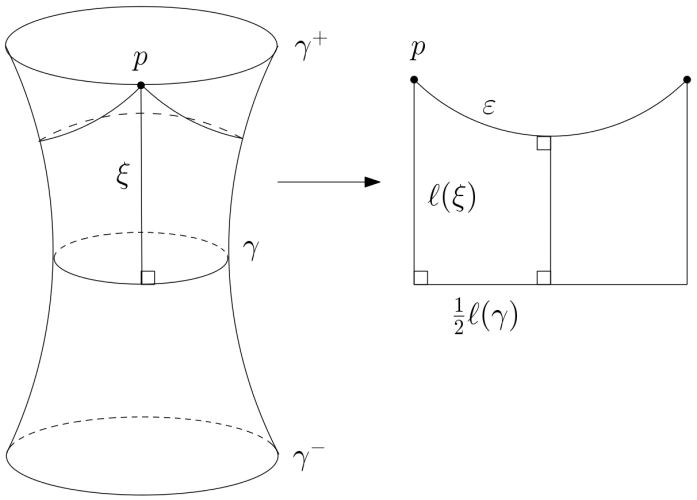


Figure 4 Computing K_C.

We see that γ^+ and, by symmetry, γ^- consist of points that are equidistant to γ with distance K_C. Moreover, γ^+ and γ^- are smooth.

Recall the notion of a collar from Section 2. In particular, each hyperbolic cylinder C in $X^{\varepsilon}_{\text{thin}}$ is a collar of width K_C, i.e., $C = C_\gamma(K_C)$. Comparing equation (2) for K_C with equation (1) in the statement of the Collar Lemma, we see that $w(\gamma) > K_C$, because $\sinh \varepsilon < 1$. This inequality will be used in the proof of Lemma 7 to give a lower bound for the distance between distinct hyperbolic cylinders in $X^{\varepsilon}_{\text{thin}}$.

We distinguish between two kinds of hyperbolic cylinders in $X^{\varepsilon}_{\text{thin}}$, namely ε'-thin cylinders and ε'-thick cylinders, where $\varepsilon' = 0.99\varepsilon$. An ε'-thick cylinder with waist γ satisfies $2\varepsilon' \leq \ell(\gamma) \leq 2\varepsilon$, since γ is contained in the ε-thin part. An ε'-thin cylinder satisfies $\ell(\gamma) < 2\varepsilon'$.

Lemma 14 in Section 3.2 states that the triangulation depicted in Figure 5 is a Delaunay triangulation for ε'-thin cylinders. We call this triangulation a *standard triangulation* and describe it in more detail in the following definition. For ε'-thick cylinders we use a different construction defined in Definition 10.

▶ **Definition 6.** *Let X be a closed hyperbolic surface. Let C be an ε'-thin hyperbolic cylinder in $X^{\varepsilon}_{\text{thin}}$ with waist γ and boundary curves γ^+, γ^-. Place three equally-spaced points $x_i, i = 1, 2, 3$ on γ (see Figure 5). Then, place three points $x_i^+, i = 1, 2, 3$ on γ^+ and three points $x_i^-, i = 1, 2, 3$ on γ^- such that the projection of $x_i^{\pm}$ on γ is equal to x_i for $i = 1, 2, 3$. Let V be the set consisting of x_i, x_i^- and x_i^+ for $i = 1, 2, 3$. Let E be the set of edges of one of the forms $(x_i^-, x_{i+1}^-), (x_i^-, x_i), (x_i^-, x_{i+1}), (x_i, x_{i+1}), (x_i, x_i^+), (x_i, x_{i+1}^+), (x_i^+, x_{i+1}^+)$ for $i = 1, 2, 3$ (counting modulo 3), where the embedding of an edge in C is as shown in Figure 5. We call (V, E) a* standard triangulation *of C.*

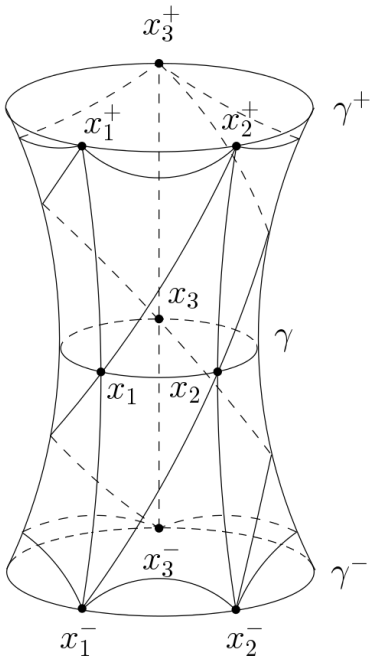


Figure 5 Standard triangulation of an ε'-thin cylinder.

We not only have to prove that a standard triangulation of an ε'-thin cylinder is a Delaunay triangulation, we also have to show that its edges are distance paths. Corollary 9 states that all edges in a standard triangulation are distance paths if $\varepsilon \leq 0.72$. Before we can prove Corollary 9, we first need the following lemma.

▶ **Lemma 7.** *Let X be a closed hyperbolic surface and let $\varepsilon \leq 0.72$. For each pair of distinct closed geodesics γ_1 and γ_2 in $X^{\varepsilon}_{\text{thin}}$ the collars $C_{\gamma_1}(K_{C_1} + \frac{1}{3}\varepsilon)$ and $C_{\gamma_2}(K_{C_2} + \frac{1}{3}\varepsilon)$ are embedded and disjoint.*

▶ Remark 8. The value 0.72 was found experimentally and is optimal up to two decimal digits, i.e., the statement is not true for $\varepsilon = 0.73$. More specifically, if $\varepsilon \geq 0.73$ then there exists a closed hyperbolic surface X with disjoint closed geodesics γ_1 and γ_2 in $X^{\varepsilon}_{\text{thin}}$ such that $C_{\gamma_1}(K_{C_1} + \frac{1}{3}\varepsilon)$ and $C_{\gamma_2}(K_{C_2} + \frac{1}{3}\varepsilon)$ are not disjoint.

Proof. See Figure 6. We will show that $w(\gamma_i) - K_{C_i} \geq \frac{1}{3}\varepsilon$ for $i = 1, 2$. Namely, this implies that $C_{\gamma_i}(K_{C_i} + \frac{1}{3}\varepsilon) \subseteq C_{\gamma_i}(w(\gamma_i))$. Because $C_{\gamma_1}(w(\gamma_1))$ and $C_{\gamma_2}(w(\gamma_2))$ are embedded and disjoint by the Collar Lemma, it follows that $C_{\gamma_1}(K_{C_1} + \frac{1}{3}\varepsilon)$ and $C_{\gamma_2}(K_{C_2} + \frac{1}{3}\varepsilon)$ are embedded and disjoint as well. Comparing expression (2) for K_{C_i} and expression (1) for $w(\gamma_i)$, we see that $w(\gamma_i) - K_{C_i}$ is a positive number, with infimum when $\ell(\gamma_i) \to 0$ [14]. A straightforward computation shows that for $\varepsilon = 0.72$ this infimum is equal to $0.24\ldots > \frac{1}{3}\varepsilon$. Since $w(\gamma_i) - K_{C_i}$ is decreasing as a function of ε, it follows that $w(\gamma_i) - K_{C_i} \geq \frac{1}{3}\varepsilon$ for all $\varepsilon \leq 0.72$. ◀

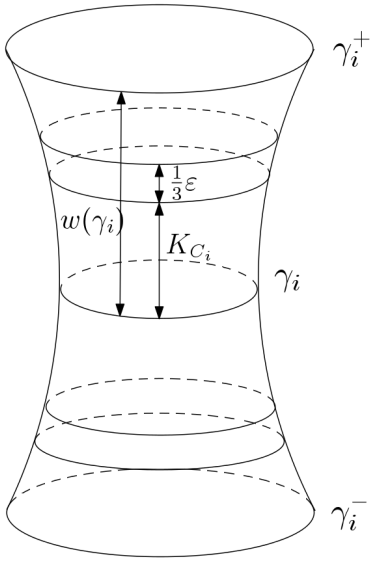


Figure 6 Illustration of the collars $C_{\gamma_i}(K_{C_i}) \subset C_{\gamma_i}(K_{C_i} + \frac{1}{3}\varepsilon) \subseteq C_{\gamma_i}(w(\gamma_i))$.

▶ **Corollary 9** (Proof in full version)**.** *Let X be a closed hyperbolic surface and let $\varepsilon \leq 0.72$. All edges in a standard triangulation of an ε'-thin cylinder in $X^{\varepsilon}_{\text{thin}}$ are distance paths.*

For ε'-thick cylinders, we see from Equation (2) for K_C that the width K_C is close to zero. It turns out that we only need to place three points on its waist.

▶ **Definition 10.** *Let X be a closed hyperbolic surface. Let C be a ε'-thick hyperbolic cylinder in $X^{\varepsilon}_{\text{thin}}$ with waist γ. Place three equally-spaced points $x_i, i = 1, 2, 3$ on γ. Let $V = \{x_i \mid i = 1, 2, 3\}$ and $E = \{(x_1, x_2), (x_2, x_3), (x_3, x_1)\}$. We call (V, E) a* standard cycle *of C.*

3.2 Constructing a distance Delaunay triangulation of X with few vertices

After placing points in $X^{\varepsilon}_{\text{thin}}$, we construct a sufficiently dense and well-distributed set of vertices in the remainder of the surface. The following definition shows more precisely how we construct a set of vertices in $X^{\varepsilon}_{\text{thick}}$ and a corresponding Delaunay triangulation.

▶ **Definition 11.** *Set $\varepsilon = 0.72$ and $\varepsilon' = 0.99\varepsilon$. Let X be a closed hyperbolic surface. Let $\mathcal{P}_1$ be the set consisting of the vertices of a standard triangulation of every ε'-thin cylinder in $X^{\varepsilon}_{\text{thin}}$ together with the vertices of a standard cycle for every ε'-thick cylinder in $X^{\varepsilon}_{\text{thin}}$. Let T_j be the union of triangles in a standard triangulation (V_j, E_j) of an ε'-thin cylinder C_j. For every ε'-thick cylinder C_j, set $T_j = \emptyset$. Define $\mathcal{P}_2$ to be a maximal set in $X \setminus \cup_j T_j$ such that $d(p, q) \geq \frac{1}{2}\varepsilon$ for all distinct $p \in \mathcal{P}_1 \cup \mathcal{P}_2, q \in \mathcal{P}_2$. Denote the union $\mathcal{P}_1 \cup \mathcal{P}_2$ by $\mathcal{P}$ and let T be the Delaunay triangulation of $\mathcal{P}$ on X obtained after projecting a Delaunay triangulation of $\pi^{-1}(\mathcal{P})$ in $\mathbb{D}$ to X. We call T a* thick-thin Delaunay triangulation *of X. The vertices in $\mathcal{P}_1$ and $\mathcal{P}_2$ are called the* cylinder vertices *and* non-cylinder vertices *of T, respectively.*

▶ Remark 12. Because by Corollary 9 all edges in a standard triangulation of any ε'-thin cylinder are distance paths if we choose $\varepsilon \leq 0.72$, we have chosen $\varepsilon = 0.72$ in Definition 11. Namely, we will see in the proof of Theorem 4 that the larger we choose ε, the smaller the constant (in our case 151) in the upper bound for the number of vertices. As in Section 3.1 we will fix $\varepsilon = 0.72$ and $\varepsilon' = 0.99\varepsilon$ throughout this subsection.

The edges between vertices on the same boundary curve of C_j are not equal to the boundary curves of C_j (because the latter are not geodesics), so T_j is strictly contained in C_j. We define $\mathcal{P}_2$ as a point set in $X \setminus \cup_j T_j$ instead of in $X \setminus \cup_j C_j$ to simplify our proof of Lemma 16, where we show that a thick-thin Delaunay triangulation of X is a simplicial complex. The definition of $\mathcal{P}$ does not explicitly forbid placing vertices of $\mathcal{P}_2$ in ε'-thick cylinders. However, we will see in the next lemma that there are no vertices of $\mathcal{P}_2$ in ε'-thick cylinders, because then they would be too close to the vertices of a standard cycle.

▶ **Lemma 13** (Proof in full version). *Let X be a closed hyperbolic surface and let T be a thick-thin Delaunay triangulation of X. Every vertex of T contained in an ε'-thick cylinder in $X^{\varepsilon}_{\text{thin}}$ is a cylinder vertex.*

Even though the set of vertices of a thick-thin Delaunay triangulation of X contains the vertices of a standard triangulation (V_j, E_j) for every ε'-thin cylinder C_j, a priori it is not clear that the edges in E_j are edges in T as well. In the next lemma, we will show that for every ε'-thin cylinder the triangles in a standard triangulation are Delaunay triangles with respect to the set of vertices of any thick-thin Delaunay triangulation of X. Namely, if this holds, then there exists a Delaunay triangulation of $\mathcal{P}$ on X containing a standard triangulation of every ε'-thin cylinder in $X^{\varepsilon}_{\text{thin}}$.

▶ **Lemma 14** (Proof in full version). *Let X be a closed hyperbolic surface. Let T be a thick-thin Delaunay triangulation of X with vertex set $\mathcal{P}$ and let C be an ε'-thin cylinder in $X^{\varepsilon}_{\text{thin}}$ with waist γ. Let (V, E) be a standard triangulation of C such that $V \subset \mathcal{P}$. Then all triangles of (V, E) are Delaunay triangles with respect to the point set $\mathcal{P}$.*

Henceforth, we will assume that for each ε'-thin cylinder the vertices and edges of a standard triangulation are contained in a thick-thin Delaunay triangulation of X. To show that $T \in \mathcal{D}(X)$, we must show that T is a simplicial complex, i.e. it does not contain any 1- or 2-cycles, and that its edges are distance paths. This is stated in Lemma 16, for which we need the following preliminary lemma.

▶ **Lemma 15** (Proof in full version). *Let X be a closed hyperbolic surface and let T be a thick-thin Delaunay triangulation of X. Any edge of T that intersects $X^{\varepsilon}_{\text{thick}}$ has length smaller than ε and is a distance path. Moreover, there are no 1- or 2-cycles that intersect $X^{\varepsilon}_{\text{thick}}$ and consist of edges of length smaller than ε.*

▶ **Lemma 16** (Proof in full version). *Every thick-thin Delaunay triangulation of a closed hyperbolic surface is a distance Delaunay triangulation.*

3.3 Proof of Theorem 4

Proof (Theorem 4). Let X be an arbitrary hyperbolic surface of genus g and let T be a thick-thin Delaunay triangulation of X. By definition, T is a Delaunay triangulation. By Lemma 16, T is a simplicial complex and all edges of T are distance paths. Hence, $T \in \mathcal{D}(X)$.

We will show here that the number of vertices of T is smaller than $151g$. By Lemma 5, $X^{\varepsilon}_{\text{thin}}$ consists of at most $3g-3$ cylinders and each of these cylinders contains either 9 vertices (if it is ε'-thin) or 3 vertices (if it is ε'-thick). Therefore, $|\mathcal{P}_1| \leq 27g - 27$.

To find an upper bound for the cardinality of $\mathcal{P}_2$, observe that for distinct $p, q \in \mathcal{P}_2$ the disks $B_p(\frac{1}{4}\varepsilon)$ and $B_q(\frac{1}{4}\varepsilon)$ of radius $\frac{1}{4}\varepsilon$ centered at p and q, respectively, are embedded and disjoint. Therefore, the cardinality of $\mathcal{P}_2$ is bounded above by the number of disjoint,

embedded disks of radius $\frac{1}{4}\varepsilon$ that we can fit in X. Because the area of a hyperbolic disk of radius $\frac{1}{4}\varepsilon$ is $2\pi(\cosh(\frac{1}{4}\varepsilon) - 1)$ [1] and because the area of X is $4\pi(g-1)$ [16], we obtain

$$|\mathcal{P}_2| \leq \frac{2(g-1)}{\cosh(\frac{1}{4}\varepsilon) - 1}.$$

Combining the upper bounds for $|\mathcal{P}_1|$ and $|\mathcal{P}_2|$ and plugging in $\varepsilon = 0.72$ yields the result. ◀

▶ Remark 17. The constant 151 is not optimal. We can obtain the stronger upper bound $|\mathcal{P}| \leq 124g$ by looking more precisely at the upper bounds of $|\mathcal{P}_1|$ and $|\mathcal{P}_2|$ but because we are mainly interested in the the order of growth, we will not provide any details.

4 Classes of hyperbolic surfaces attaining the order of the upper bound

As our second result, we show that there exists a class of hyperbolic surfaces which attains the order of the upper bound presented in Theorem 4. We will first introduce this class of hyperbolic surfaces and then state the precise result in Theorem 18.

Using the idea of a trivalent graph associated to a pair of pants decomposition discussed in Section 2, define L_g as the trivalent graph depicted in Figure 7. Here, every vertex v_i corresponds to a pair of pants Y_i. There is one edge from v_1 to itself and similarly from v_{2g-2} to itself. Moreover, for $1 \leq i \leq 2g-3$ there is one edge between v_i and v_{i+1} if i is odd and there are two edges if i is even.

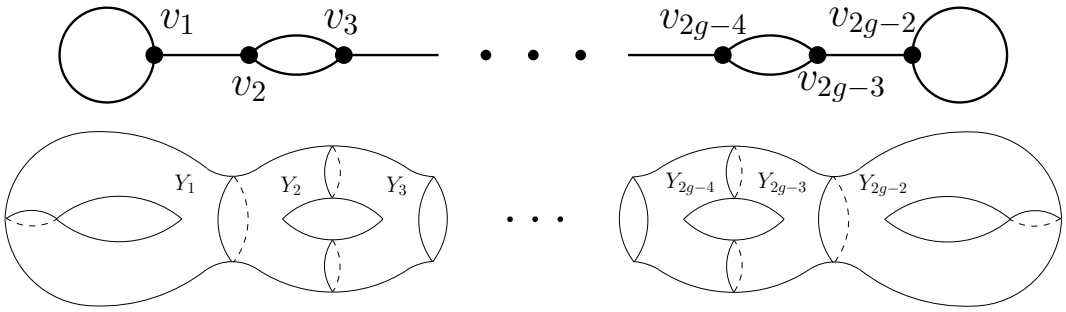


Figure 7 Trivalent graph L_g (top) with corresponding pair of pants decomposition (bottom).

Now, fix some interval $[a, b] \subset \mathbb{R}$ with $0 < a < b$. Let $S_g(a, b)$ be the subset of $\mathcal{T}_g$ with underlying graph L_g such that all length parameters are contained in $[a, b]$. In particular, $S_g(a, b)$ contains an open subset of $\mathcal{T}_g$. The following result thus shows that having a linear number of vertices in terms of the genus is relatively stable in this part of Teichmüller space.

▶ **Theorem 18.** *There exists a constant $B > 0$ depending only on a, b such that a minimal distance Delaunay triangulation of any hyperbolic surface in $S_g(a, b)$ has at least Bg vertices.*

The idea of the proof is the following. Let a hyperbolic surface $X \in S_g(a, b)$ and a triangulation $T \in \mathcal{D}(X)$ be given. Euler's formula implies $v - \frac{1}{3}e = 2 - 2g$ for triangulations of a surface of genus g, where v and e are the number of vertices and edges of the triangulation. We prove that $e \leq B'v$ for some constant $B' > 3$ only depending on a, b, which implies that

$$v \geq \frac{6g-6}{B'-3}.$$

This implies the result of Theorem 18. Hence, the argument consists mostly in finding an upper bound for the number of edges in terms of the number of vertices.

Before we continue with the proof of Theorem 18, we will look at our construction of $S_g(a,b)$ in more detail. By definition, every boundary geodesic of a pair of pants in the pair of pants decomposition of $X \in S_g(a,b)$ with respect to L_g has length in $[a,b]$. As explained in Section 2, the geometry of a pair of pants depends continuously on the lengths of its three boundary geodesics. In particular, the diameter $\operatorname{diam}(Y)$ of a pair of pants Y as well as the minimal distance $\operatorname{mindist}(Y)$ between any two of its boundary geodesics depend continuously on the lengths of its boundary geodesics. Because $[a,b]$ is a compact set, we obtain as an immediate consequence the following lemma.

▶ **Lemma 19.** *There exist positive numbers $m(a,b)$ and $M(a,b)$ depending on a and b such that $m(a,b) \leq \operatorname{mindist}(Y) < \operatorname{diam}(Y) \leq M(a,b)$ for every pair of pants Y whose boundary geodesics have length in $[a,b]$.*

▶ Remark 20. It is not too difficult to compute bounds for $\operatorname{mindist}(Y)$ and $\operatorname{diam}(Y)$ in terms of the lengths of the boundary geodesics of Y. This would give explicit expressions for $m(a,b)$ and $M(a,b)$ in terms of a and b. As we are only interested in the order of growth, to avoid further technical details, we do not provide details.

From now on, the numbers $m = m(a,b)$ and $M = M(a,b)$ will be fixed. Furthermore, a *cluster* in a hyperbolic surface X is a subset of X consisting of a number of consecutive pairs of pants, where *consecutive* is with respect to the ordering of L_g. Consider $T \in \mathcal{D}(X)$. A *k-gap* is a cluster consisting of k consecutive empty pairs of pants, where *empty* means that the pairs of pants do not contain any vertices of T. If a vertex of T is contained in two pairs of pants, i.e., if the vertex lies on a boundary geodesic, then we only count it as a vertex of the pair of pants with the lowest index in L_g. We say that an edge of T *crosses* a cluster if the pairs of pants containing its endpoints are separated by the cluster. Note that the cluster need not contain all pairs of pants which separate the two endpoints.

The next lemma states that if an edge of a distance Delaunay triangulation crosses many pairs of pants, then "many" of these pairs of pants are empty.

▶ **Lemma 21** (Proof in full version)**.** *Let $X \in S_g(a,b)$ and define $N = N(a,b)$ as*

$$N(a,b) := \left\lceil \frac{M(a,b)}{m(a,b)} \right\rceil + 1.$$

Then, for every $T \in \mathcal{D}(X)$ the following statements hold:

1. *If an edge of T crosses a cluster consisting of at least $3N$ pairs of pants, this cluster contains an N-gap.*
2. *If an edge of T crosses a cluster in which the first N and the last N pairs of pants are empty, then all pairs of pants in the cluster are empty.*

The following lemma states that we can construct a set of clusters which has as one of its properties that every edge of the distance Delaunay triangulation has its endpoints in the same cluster, or in two consecutive clusters.

▶ **Lemma 22** (Proof in full version)**.** *Let $X \in S_g(a,b)$ be a hyperbolic surface and let $N = N(a,b)$ be as defined in Lemma 21. Let $T \in \mathcal{D}(X)$. There are interior-disjoint clusters with the following properties:*

1. *Each cluster consists of at most $6N$ consecutive pairs of pants;*
2. *Every cluster contains at least one vertex of $\mathcal{T}$, and every vertex of $\mathcal{T}$ belongs to exactly one cluster;*
3. *Every edge of $\mathcal{T}$ has its endpoints in the same cluster, or in two consecutive clusters.*

In the following corollary, we denote the number of vertices of $T \in \mathcal{D}(X)$ contained in a subset U of X by $v(U)$. Likewise, let $e(U, W)$ be the number of edges of T with one endpoint in $U \subset X$ and one endpoint in $W \subset X$.

▶ **Corollary 23.** *Let $X \in S_g(a, b)$ be a hyperbolic surface and let $T \in \mathcal{D}(X)$. Let $\{\Gamma_i \mid i = 1, \ldots, n\}$ be a collection of clusters satisfying the properties of Lemma 22 for some $n \in \mathbb{N}$. If v and e are the number of vertices and edges of T, respectively, then*

$$n \leq v,$$
$$v = \sum_{i=1}^{n} v(\Gamma_i),$$
$$e = \sum_{i=1}^{n} e(\Gamma_i, \Gamma_i) + \sum_{i=1}^{n-1} e(\Gamma_i, \Gamma_{i+1}).$$

Proof. Because every cluster contains at least one vertex, the number of clusters is at most the number of vertices, which proves the first equation. The second equation follows from the property that every vertex is contained in a cluster. Because every edge has its endpoints in the same cluster, or in two consecutive clusters, the third equation holds. ◀

Recall that we want to find a linear upper bound for the number of edges of a distance Delaunay triangulation in terms of the number of vertices. By Corollary 23, it suffices to find upper bounds for $e(\Gamma_i, \Gamma_i)$ and $e(\Gamma_i, \Gamma_{i+1})$ for clusters Γ_i satisfying the properties of Lemma 22. We will do this in the next lemma.

▶ **Lemma 24** (Proof in full version)**.** *With notation as in Corollary 23, the following upper bounds hold:*

1. $e(\Gamma_i, \Gamma_i) \leq 3v(\Gamma_i) + 18N(N+1)$ *for all* $i = 1, \ldots, n$,
2. $e(\Gamma_i, \Gamma_{i+1}) \leq 18v(\Gamma_i \cup \Gamma_{i+1}) + 216N(N+1)$ *for all* $i = 1, \ldots, i-1$.

We can now commence with the proof of Theorem 18.

Proof (Theorem 18). Take $X \in S_g(a, b)$ arbitrary and let $T \in \mathcal{D}(X)$ be arbitrary. Let $\{\Gamma_i \mid i = 1, \ldots, n\}$ be a collection of clusters satisfying the properties of Lemma 22. By Corollary 23,

$$e = \sum_{i=1}^{n} e(\Gamma_i, \Gamma_i) + \sum_{i=1}^{n-1} e(\Gamma_i, \Gamma_{i+1}).$$

Substituting the upper bounds for $e(\Gamma_i, \Gamma_i)$ and $e(\Gamma_i, \Gamma_{i+1})$ from Lemma 24, we obtain

$$e \leq 39 \sum_{i=1}^{n} \big(v(\Gamma_i) + 6N(N+1)\big).$$

From Corollary 23, we know that $\sum_{i=1}^{n} v(\Gamma_i) = v$ and $n \leq v$. Hence, $e \leq 39(1+6N(N+1))v$. Euler's formula for triangulations $v - \frac{1}{3}e = 2 - 2g$ implies that

$$39(1 + 6N(N+1))v \geq e = 3v + 6g - 6,$$
$$v \geq \frac{g-1}{6 + 39N(N+1)},$$

which finishes the proof. ◀

5 Lower bound

In this section, we will look at a general lower bound for the minimal number of vertices of a distance Delaunay triangulation of a hyperbolic surface of genus g.

In the more general situation of a simplicial triangulation of a topological surface of genus g, one has an immediate lower bound on the minimal number of vertices. The fact that this lower bound is sharp is the following classical theorem of Jungerman and Ringel:

▶ **Theorem 25** ([12, Theorem 1.1])**.** *The minimal number of vertices of a simplicial triangulation of a topological surface of genus g is*

$$\left\lceil \frac{7+\sqrt{1+48g}}{2} \right\rceil.$$

We show that the same result holds for the minimal number of vertices of a distance Delaunay triangulation of a hyperbolic surface of genus g for infinitely many values of g.

▶ **Theorem 26.** *For any $g \geq 2$ of the form $g = \frac{1}{12}(n-3)(n-4)$ for some $n \equiv 0 \mod 12$, the minimal number of vertices of a distance Delaunay triangulation of a hyperbolic surface of genus g is*

$$n = \frac{7+\sqrt{1+48g}}{2}.$$

Proof. Because there are no distance Delaunay triangulations with fewer than the stated number of vertices by Theorem 25, it is sufficient to construct for a given hyperbolic surface a distance Delaunay triangulation with the stated number of vertices.

Our construction is inspired by a similar construction in the context of the chromatic number of hyperbolic surfaces [14]. Let $n \equiv 0 \mod 12$ and assume that $n \neq 0$. The complete graph K_n on n vertices can be embedded in a topological surface S_g of genus $g = \frac{1}{12}(n-3)(n-4)$ which is the smallest possible genus [15]. Because we have assumed that $n \equiv 0 \mod 12$, we know that the embedding of K_n into S_g is a triangulation T [18]. To turn T into a distance Delaunay triangulation, we will add a hyperbolic metric to the topological surface as follows. Every triangle in T is replaced by the unique equilateral hyperbolic triangle with all three angles equal to $\frac{2\pi}{n-1}$. In the complete graph K_n every vertex has $n-1$ neighbouring vertices. This means that in every vertex $n-1$ equilateral triangles meet, so the total angle at each vertex is 2π. Therefore, the result after replacing all triangles in T by hyperbolic triangles is a smooth hyperbolic surface Z_g.

It remains to be shown that $T \in \mathcal{D}(Z)$. By construction, T is a simplicial complex. It has also been shown that all edges are distance paths [14]. We will show here that T is a Delaunay triangulation of Z_g. Consider an arbitrary triangle (u,v,w) in T with circumcenter c and let $p \notin \{u,v,w\}$ be an arbitrary vertex of T (Figure 8). Consider a distance path γ from c to p. We can regard γ as the concatenation of simple segments that each pass through an individual triangle.

The first of these simple segments starts from c and leaves the triangle (u,v,w), so its length is at least the distance between c and a side of (u,v,w). Therefore, denoting by x the projection of c on one of the edges as shown in Figure 8, the length of the first segment is at least $d(c,x)$. The last of the simple segments passes through a triangle, say Δ, before arriving at p, so it has to pass through the side of Δ opposite to p. Therefore, its length is at least the distance between p and the opposite side of Δ. It is known that the distance between a vertex and the opposite side of an equilateral triangle is at least $\frac{1}{2}\ell$, where ℓ denotes the length of the sides of the equilateral triangle [14]. Hence, $d(c,p) = \ell(\gamma) \geq d(c,x) + \frac{1}{2}\ell$. By the

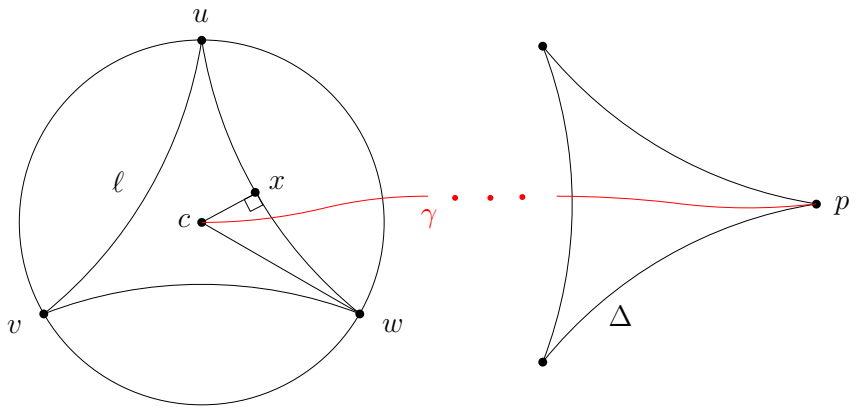


Figure 8 Schematic overview of the proof of T being a Delaunay triangulation.

triangle inequality in triangle (c, w, x) we see that $d(c, w) \leq d(c, x) + d(x, w) = d(c, x) + \frac{1}{2}\ell$, so we conclude that $d(c, p) \geq d(c, w)$. This means that p is not contained in the interior of the circumcircle of (u, v, w), which shows that (u, v, w) is a Delaunay triangle. By symmetry, it follows that all triangles are Delaunay triangles, which finishes the proof. ◀

References

1 Alan F. Beardon. *The geometry of discrete groups*, volume 91 of *Graduate Texts in Mathematics*. Springer-Verlag, 2012.

2 Lipman Bers. An inequality for Riemann surfaces. In *Differential geometry and complex analysis*, pages 87–93. Springer, Berlin, 1985.

3 Mikhail Bogdanov and Monique Teillaud. Delaunay triangulations and cycles on closed hyperbolic surfaces. Technical Report RR-8434, INRIA, 2013.

4 Mikhail Bogdanov, Monique Teillaud, and Gert Vegter. Delaunay triangulations on orientable surfaces of low genus. In Leibniz International Proceedings in Informatics, editor, *32nd International Symposium on Computational Geometry (SoCG 2016)*, pages 20:1–20:17, 2016.

5 Adrian Bowyer. Computing Dirichlet tessellations. *The Computer Journal*, 24(2):162–166, 1981.

6 Peter Buser. *Geometry and spectra of compact Riemann surfaces*. Springer-Verlag, 2010.

7 Vincent Despré, Jean-Marc Schlenker, and Monique Teillaud. Flipping geometric triangulations on hyperbolic surfaces. *arXiv preprint*, 2019. `arXiv:1912.04640`.

8 István Fáry. On straight-line representation of planar graphs. *Acta scientiarum mathematicarum*, 11(229-233):2, 1948.

9 Larry Guth, Hugo Parlier, and Robert Young. Pants decompositions of random surfaces. *Geom. Funct. Anal.*, 21(5):1069–1090, 2011.

10 Alfredo Hubard, Vojtěch Kaluža, Arnaud De Mesmay, and Martin Tancer. Shortest path embeddings of graphs on surfaces. *Discrete & Computational Geometry*, 58(4):921–945, 2017.

11 Iordan Iordanov and Monique Teillaud. Implementing Delaunay triangulations of the Bolza surface. In *Proceedings of the Thirty-third International Symposium on Computational Geometry*, pages 44:1–44:15, 2017.

12 Mark Jungerman and Gerhard Ringel. Minimal triangulations on orientable surfaces. *Acta Mathematica*, 145(1):121–154, 1980.

13 Maryam Mirzakhani. Growth of Weil-Petersson volumes and random hyperbolic surfaces of large genus. *J. Differential Geom.*, 94(2):267–300, 2013.

14 Hugo Parlier and Camille Petit. Chromatic numbers of hyperbolic surfaces. *Indiana University Mathematics Journal*, pages 1401–1423, 2016.

15 Gerhard Ringel and John W.T. Youngs. Solution of the Heawood map-coloring problem. *Proceedings of the National Academy of Sciences*, 60(2):438–445, 1968.

16 John Stillwell. *Geometry of surfaces*. Springer-Verlag, 1992.

17 Roberto Tamassia. *Handbook of graph drawing and visualization*. Chapman and Hall/CRC, 2013.

18 Charles M. Terry, Lloyd R. Welch, and John W.T. Youngs. The genus of K_{12s}. *Journal of Combinatorial Theory*, 2(1):43–60, 1967.

The Density Fingerprint of a Periodic Point Set

Herbert Edelsbrunner
IST Austria (Institute of Science and Technology Austria), Klosterneuburg, Austria

Teresa Heiss
IST Austria (Institute of Science and Technology Austria), Klosterneuburg, Austria

Vitaliy Kurlin
Department of Computer Science, University of Liverpool, UK

Philip Smith
Department of Computer Science, University of Liverpool, UK

Mathijs Wintraecken
IST Austria (Institute of Science and Technology Austria), Klosterneuburg, Austria

Abstract

Modeling a crystal as a periodic point set, we present a fingerprint consisting of density functions that facilitates the efficient search for new materials and material properties. We prove invariance under isometries, continuity, and completeness in the generic case, which are necessary features for the reliable comparison of crystals. The proof of continuity integrates methods from discrete geometry and lattice theory, while the proof of generic completeness combines techniques from geometry with analysis. The fingerprint has a fast algorithm based on Brillouin zones and related inclusion-exclusion formulae. We have implemented the algorithm and describe its application to crystal structure prediction.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Lattices, periodic sets, isometries, Dirichlet–Voronoi domains, Brillouin zones, bottleneck distance, stability, continuity, crystal database

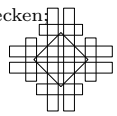
Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.32

Funding *Herbert Edelsbrunner*: ERC Horizon 2020 "Alpha Shape Theory Extended", no. 788183; FWF "Wittgenstein Prize", no. Z 342-N31; FWF DFG TRR 109 "Discretization in Geometry and Dynamics", no. I 02979-N35.
Teresa Heiss: ERC Horizon 2020 "Alpha Shape Theory Extended", no. 788183.
Vitaliy Kurlin: EPSRC grant "Application-driven Topological Data Analysis" (EP/R018472/1).
Philip Smith: Leverhulme Research Centre for Functional Materials Design at the University of Liverpool, UK.
Mathijs Wintraecken: the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 754411.

Acknowledgements The authors thank Janos Pach for insightful discussions on the topic of this paper, Morteza Saghafian for finding the one-dimensional counterexample mentioned in Section 5, and Larry Andrews for generously sharing his crystallographic perspective.

1 Introduction

This paper considers a deceptively simple question: *given periodic point sets (crystals) in* $\mathbb{R}^3$*, determine how close the sets are to being isometric.* In other words, how much do the points need to be perturbed to allow for a rigid transformation between the two sets? More generally, we may ask for the organization of a collection of periodic sets that facilitates efficient search. In this context, a *periodic (point) set* is the Minkowski sum of a *lattice* and a *motif*. The lattice is spanned by three linearly independent vectors, and the motif is a finite



set of points in the *unit cell*, which is the parallelepiped whose edges are translates of the three vectors. The main reason for the difficulty of the question is the complicated nature of the continuous space of isometry classes of periodic sets:

- There is no method for choosing a unique basis for a lattice in a continuous manner. Indeed, continuity contradicts uniqueness as we can continuously deform a basis to a different basis of the same lattice. For example, the Niggli reduced cell [15] is unique but not continuous with respect to perturbations of the lattice [1].
- Crystallographers often use the symmetry group of crystals, which define a stratification of the space of isometry classes. Belonging to a given stratum is however not a continuous property.
- Small perturbations of a periodic set can significantly change the lattice with respect to which it is periodic.

Periodic sets are usually given by a basis of a lattice and the motif within the unit cell spanned by this basis. As explained above, even for extremely similar periodic sets, their lattices, their bases, and thus their motifs can look completely different, making a direct comparison between the motifs impossible. It is therefore advisable not to compute the distances between the isometry classes but instead map the sets to a less complicated metric space. This is the approach we take in this paper. Part of the challenge is to determine which properties this map should possess, and how to balance its mathematical properties with efficient computability. As in many applications, we consider false positives in the comparison of two crystals less problematic than false negatives: a large distance in the metric space should imply that the two periodic sets are far from isometric, while a small distance should indicate a high chance that the sets are indeed close to being isometric.

The main contribution of this paper is a candidate solution, which we refer to as the *density fingerprint map*. For different non-negative integers, k, and for different non-negative radii, t, it maps the periodic set to the probability that a random point in the unit cell is at distance at most t from exactly k points in the periodic set; see Definition 3.1.

▶ **Main Theorem.** *The density fingerprint maps a periodic set in $\mathbb{R}^3$ to a series of density functions that satisfy the following properties:*

1. *the map is invariant under isometries (rigid motions and reflections) of space;*
2. *the map is Lipschitz continuous with respect to small perturbations of the points, with the Lipschitz constant depending on the packing and covering radii of the periodic set;*
3. *the map is generically complete: for a dense and open subset of the space of periodic sets, the isometry class of the periodic set is uniquely determined.*

Short of proving completeness beyond the generic case, we leave the completeness of the density fingerprint map for all periodic sets as an open question. Indeed, the authors of this paper failed to produce a counterexample to general completeness, but not because of a lack of trying. The Main Theorem generalizes to arbitrary finite dimensions.

The quest for a fingerprint map is motivated by the study of crystals, for which the configuration of atoms is important for their chemical properties. Quantifying the similarity between crystals has the potential to greatly improve the practice of *Crystal Structure Prediction*, which has come to rely on high-performance computing for simulating millions of structures. Many of them are similar to each other, and very few are eventually produced in the laboratory. An effective notion of similarity would allow an improved organization of the structures and lead to vast savings of supercomputing time currently wasted on redundant

simulations. The prior work in this area is best summarized by listing the software systems currently used in practice: COMPACK [5], MERCURY [13], and COMPSTRU [10]. These systems are of great help in comparing crystals, but they employ heuristics like cut-offs and tolerances, which come with the usual drawbacks. It is our ambition to develop the mathematical and computational foundations needed to overcome the current deficiencies.

Outline. Section 2 provides the necessary notation and terminology for lattices, periodic sets, and isometries. Section 3 introduces the density functions for a periodic set and the corresponding density fingerprint map. Section 4 proves that the density fingerprint map is continuous with respect to perturbations of the periodic set. Section 5 proves that the density fingerprint map is complete for generic periodic sets. Section 6 explains how the density fingerprint is computed using the Brillouin zones of the points. Section 7 describes a preliminary application of the density fingerprint map to Crystal Structure Prediction. Section 8 concludes the paper.

2 Background

We cover two topics: locally finite point sets modeling crystals and transformations between them.

2.1 Delone Sets and Periodic Sets

We recall that $A \subseteq \mathbb{R}^3$ is *locally finite* if any compact subset of $\mathbb{R}^3$ contains only finitely many points of A. It is a *Delone set* [6] if there exist $r, R > 0$ such that every open ball of radius r contains at most one point of A and every closed ball of radius R contains at least one point of A. In other words, no two points of A can be closer than $2r$ and no point of $\mathbb{R}^3$ can be further from A than R. We refer to the largest such r as the *packing radius* and the smallest such R as the *covering radius* of A. A Delone set is necessarily infinitely large and its points are in a sense evenly spread out over the entire Euclidean space.

We get an important subclass starting with three linearly independent vectors, $v_1, v_2, v_3 \in \mathbb{R}^3$, which we call a *basis*. The set of integer combinations is the *lattice*, Λ, and the set of real combinations with coefficients in $[0, 1)$ is the *unit cell*, U, the vectors span:

$$\Lambda = \{n_1v_1 + n_2v_2 + n_3v_3 \mid n_1, n_2, n_3 \in \mathbb{Z}\}, \tag{1}$$

$$U = \{r_1v_1 + r_2v_2 + r_3v_3 \mid 0 \leq r_1, r_2, r_3 < 1\}. \tag{2}$$

We call any finite set $M \subseteq U$ a *motif* and $M + \Lambda = \{x + v \mid x \in M, v \in \Lambda\}$ a *periodic (point) set*. By construction, $M + \Lambda + v = M + \Lambda$ for every $v \in \Lambda$, and if this exhausts all translations that keep $M + \Lambda$ invariant, then U is a *primitive unit cell* of $M + \Lambda$. Since the International Union of Crystallography (IUCr) allows lattices that are spanned by fewer than three linearly independent vectors, it calls lattices as defined above *full*. We observe that $M + \Lambda$ is a Delone set if Λ is full and not if Λ is not full, and therefore we will be exclusively interested in full lattices and so will assume as much without mentioning the term.

It is important to keep in mind that the basis and therefore the primitive unit cell are not unique. This is illustrated in Figure 1, which shows three of the infinitely many bases of the hexagonal lattice: a together with $b - a$, b, or $b + a$. Applying Niggli's algorithm for the *Niggli reduced cell* [15] to this particular lattice, there is an ambiguity between the bases $\{a, b\}$ and $\{a, b - a\}$, because the projections of b and $b - a$ onto the line of a both have length $\frac{1}{2}\|a\|$. The tie can be broken by preferring b, but this causes a discontinuity in the construction of the reduced unit cell.

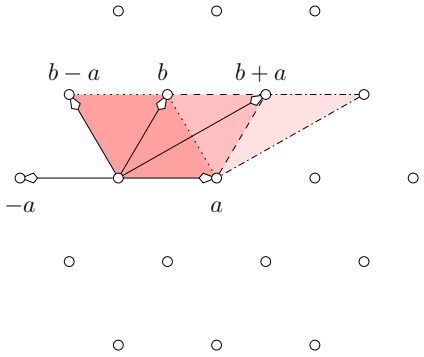


Figure 1 The bases $\{a, b-a\}$, $\{a, b\}$, $\{a, b+a\}$ of the hexagonal lattice, and the corresponding unit cells drawn as shaded parallelograms.

2.2 Rigid Motions and Isometries

Rather than a fixed set in $\mathbb{R}^3$, we often consider the class of sets that are equivalent under a particular type of transformation. For example, a *rigid motion* is a map $\mathbb{R}^3 \to \mathbb{R}^3$ that is composed of a rotation and a translation. It preserves distances between pairs of points as well as orientations of ordered triplets of points. An *isometry* is a rigid motion possibly composed with a reflection, and so preserves distances but not necessarily orientations. This is the most relevant group of transformations to this paper as we model crystals by isometry classes of periodic sets.

3 The Density Fingerprint and its Invariance

A continuous invariant for comparing crystals is the density, defined as the total volume of balls centered at points in the motif divided by the volume of the unit cell. To avoid the choice of radii, we grow the balls continuously and simultaneously from their centers and get a 1-dimensional function rather than a single number. There are still many periodic sets this function cannot distinguish, for example any hexagonal close packing from the face-centered cubic lattice. We therefore add information by distinguishing points covered by a different number of balls.

▶ **Definition 3.1** (Density Functions and Fingerprint). *Let $A = M + \Lambda \subset \mathbb{R}^3$ be a periodic set and write $A(t)$ for the collection of closed balls, $B(a;t)$, of radius $t \geq 0$ centered at the points $a \in A$. The* k-fold cover *of $A(t)$, denoted $\bigcup^k A(t)$, consists of all points $x \in \mathbb{R}^3$ contained in k or more of these balls. The fractional volume of the k-fold cover, $\varphi_k^A(t) = \mathrm{Vol}[U \cap \bigcup^k A(t)]/\mathrm{Vol}[U]$, is also the probability that a point chosen uniformly at random within a unit cell, U, belongs to at least k balls, and subtracting the fractional volume of the $(k+1)$-fold cover, we get the probability that the random point belongs to exactly k balls:*

$$\varphi_k^A(t) = \mathrm{Prob}[x \in B(a;t) \text{ for } k \text{ or more points } a \in A]; \tag{3}$$

$$\psi_k^A(t) = \varphi_k^A(t) - \varphi_{k+1}^A(t) = \mathrm{Prob}[x \in B(a;t) \text{ for exactly } k \text{ points } a \in A]. \tag{4}$$

We call $\psi_k^A \colon [0,\infty) \to [0,1]$ the k-th density function *of A. The* density fingerprint *of A is the vector of density functions: $\Psi(A) = (\psi_0^A, \psi_1^A, \ldots, \psi_k^A, \ldots)$, and $A \mapsto \Psi(A)$ is the* density fingerprint map.

See Figure 2, which illustrates the density functions for the hexagonal and the square lattices in $\mathbb{R}^2$. Note that the density fingerprint is an isometry invariant and that it neither depends on the lattice used to write A as a periodic set, nor on its basis.

▶ **Lemma 3.2** (Invariance under Isometries). *Let $A \subseteq \mathbb{R}^3$ be a periodic set, and let $Q \subseteq \mathbb{R}^3$ be isometric to A. Then $\Psi(A) = \Psi(Q)$.*

Proof. Let now $\text{iso}\colon \mathbb{R}^3 \to \mathbb{R}^3$ be the isometry for which $Q = \text{iso}(A)$, and note that it also maps $A(t)$ to $Q(t)$ and $\bigcup^k A(t)$ to $\bigcup^k Q(t)$ for every $k \geq 0$. It follows that $\psi_k^A(t) = \psi_k^Q(t)$, for every $k \geq 0$, and therefore $\Psi(A) = \Psi(Q)$, as claimed. ◀

While the fingerprint map is not invariant under similarities, we can write $\Psi(sA) = (\psi_0^A \circ s, \psi_1^A \circ s, \ldots, \psi_k^A \circ s, \ldots)$, in which $s(t) = st$ scales the radius. It would therefore be easy to construct a fingerprint map that is invariant under similarities, namely by normalizing the radius, e.g. by letting the radius be tr, in which r is the packing radius of A.

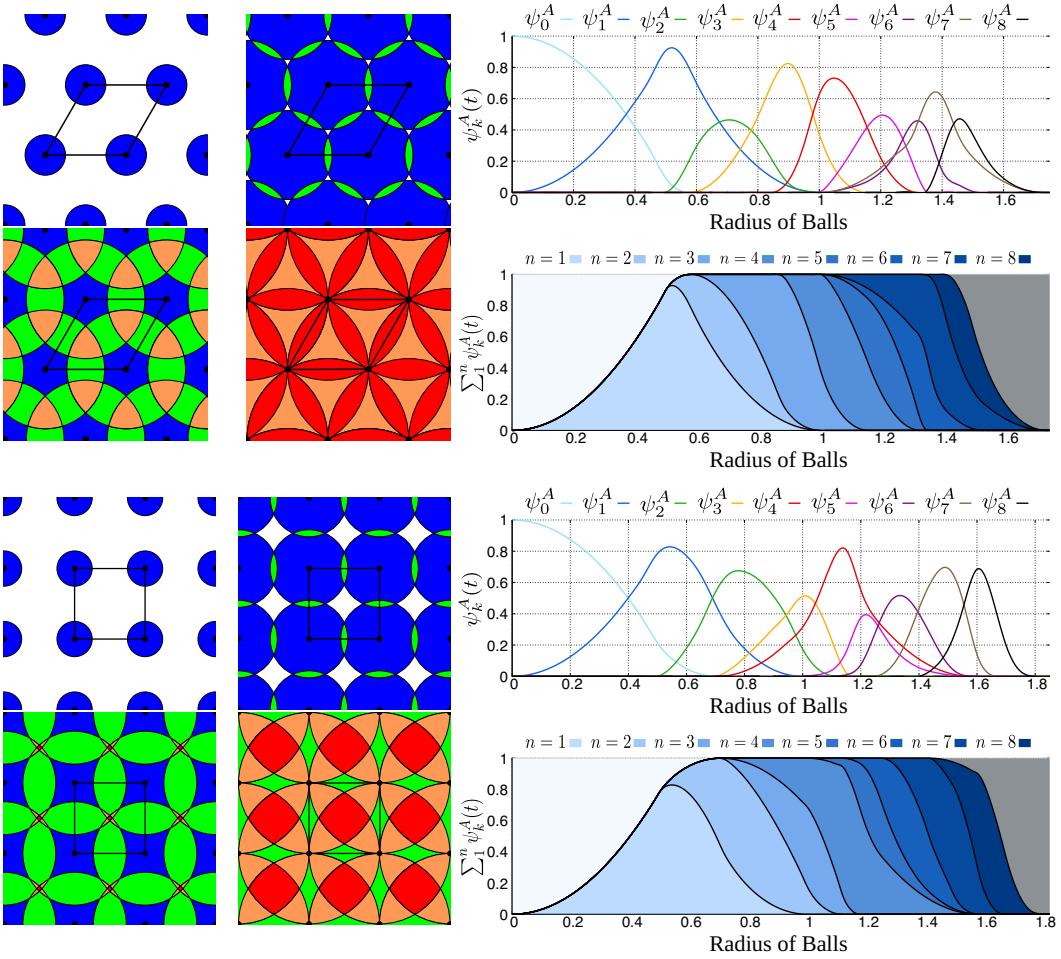


Figure 2 The density fingerprint map of the hexagonal lattice on the *top* and, for comparison, of the square lattice on the *bottom*. *Left*: the k-fold covers of the two sets for four different radii each: $t = 0.25, 0.55, 0.75, 1.00$. *Right*: the graphs of the respective first nine density functions above the corresponding *densigram*, in which the zeroth function can be seen upside-down and the remaining density functions are accumulated from left to right.

4 Continuity

We prove that the density fingerprint map is Lipschitz continuous with respect to small perturbations of the points. To formalize this result, we introduce distances between periodic sets and between density fingerprints. For sets $A, Q \subseteq \mathbb{R}^3$ of equal cardinality, the *(Euclidean) bottleneck distance* is the infimum, over all bijections, $\gamma\colon A \to Q$, of the supremum Euclidean distance between matched points, and for density fingerprints, $\Psi(A), \Psi(Q)$, we use the supremum of the weighted infinity norms of the differences between corresponding density functions:

$$d_B(A,Q) = \inf_{\gamma\colon A\to Q} \sup_{a\in A} \|a - \gamma(a)\|_2, \tag{5}$$

$$d_\infty(\Psi(A),\Psi(Q)) = \sup_{k\geq 0} \tfrac{1}{\sqrt[3]{k+1}^2} \|\psi_k^A - \psi_k^Q\|_\infty. \tag{6}$$

Note the damping of the difference between corresponding density functions. The reason for it is technical and related to the fact that density functions with higher k tend to vanish at later values of t. As a consequence, the sensitivity of the density function to any perturbation increases with growing k, and the damping compensates for this tendency. Before proving Lipschitz continuity, we show that two periodic sets with small bottleneck distance between them necessarily have a common lattice.

▶ **Lemma 4.1** (Common Lattice). *Let A, Q be periodic sets in $\mathbb{R}^3$, and let $r_Q > 0$ be the packing radius of Q. If $d_B(A,Q) < r_Q$, then there is a lattice Λ with unit cell U in $\mathbb{R}^3$ such that $\#(A \cap U) = \#(Q \cap U)$ and $A = (A \cap U) + \Lambda$ and $Q = (Q \cap U) + \Lambda$.*

Proof. Since $A, Q \subseteq \mathbb{R}^3$ are periodic, there are lattices with unit cells such that $A = (A \cap U_A) + \Lambda_A$ and $Q = (Q \cap U_Q) + \Lambda_Q$. To get a contradiction, we assume that there is however no common lattice for A and Q. Equivalently, $\Lambda_A \cap \Lambda_Q$ is a lattice of dimension at most 2. Therefore there exists a basis vector, v, of Λ_A such that $nv \in \Lambda_Q$ implies $n = 0$. Picking a point $a \in A$, we consider the infinitely many points $a(n) = a + nv$, with $n \in \mathbb{Z}$. For each $a(n)$, let $q(n)$ be the point in Λ_Q such that $a(n) \in q(n) + U_Q$, and define $b(n) = a(n) - q(n)$, which we note belongs to U_Q.

There are infinitely many pairwise different points $b(n)$ in the unit cell, and it suffices to prove that at least one is at distance larger than $\delta = d_B(A,Q)$ from all points in Q. To see this, let $b(i)$ and $b(j)$ be at distance less than $\varepsilon = r_Q - \delta$ from each other, and note that $b(i + n[j-i]) = b(i) + n[b(j) - b(i)]$, for $n \in \mathbb{Z}$, provided the point on the right-hand side of the equation belongs to U_Q. In other words, we have an entire line of points with distance less than ε between contiguous points. The gap between balls of radius δ centered at the points in Q is at least 2ε, which implies that at least one of the points on the line is outside all such balls. This contradicts the assumption that the bottleneck distance between A and Q is $\delta = r_Q - \varepsilon$. The existence of a common lattice of A and Q follows. ◀

The proof of Lipschitz continuity makes use of the common lattice of the sets before and after the perturbation. We therefore formulate the claim assuming that the bottleneck distance between the two sets is less than the packing radii.

▶ **Theorem 4.2** (Fingerprint Continuity). *Let A, Q be periodic sets in $\mathbb{R}^3$, both with packing radius at least $r > 0$ and with covering radius at most $R < \infty$. If $\delta = d_B(A,Q) < r$, then there exists a constant $C = C(r,R)$ such that $d_\infty(\Psi(A),\Psi(Q)) \leq C \cdot d_B(A,Q)$.*

Proof. By Lemma 4.1, there is a lattice, $\Lambda \subseteq \mathbb{R}^3$, that is common to both sets, A and Q, and we write U for the corresponding unit cell. Let $\gamma\colon A \to Q$ be a bijection such that $d_B(A,Q)$ is the supremum Euclidean distance between corresponding points, let k be a non-negative integer, and let t be a positive real number. We need an upper bound for

$$\left|\psi_k^A(t) - \psi_k^Q(t)\right| = \frac{\left|\mathrm{Vol}[A_t^k \cap U] - \mathrm{Vol}[Q_t^k \cap U]\right|}{\mathrm{Vol}[U]}, \tag{7}$$

in which $A_t^k = \bigcup^k A(t) \setminus \bigcup^{k+1} A(t)$ consists of all points $x \in \mathbb{R}^3$ contained in exactly k balls of $A(t)$, and similarly for Q_t^k. As a first step, we find an upper bound on the numerator, Δ, for the case in which γ is the identity except for one point, $a \in M$, which it maps to $q = \gamma(a) \in B(a;\delta)$; that is: $Q = A \setminus (a+\Lambda) \cup (q+\Lambda)$. A point $x \in \mathbb{R}^3$ is possibly covered by a different number of balls before and after the perturbation only if $x \in [B(a;t) \ominus B(q;t)] + \Lambda$, with $\ominus$ denoting the symmetric difference. Observe that this set is contained in $[B(\frac{a+q}{2}; t+\frac{\delta}{2}) \setminus B(\frac{a+q}{2}; t-\frac{\delta}{2})] + \Lambda$. Hence,

$$\Delta \leq \mathrm{Vol}[B(a;t) \ominus B(q;t)] \leq \frac{4\pi}{3}\left[\left(t+\tfrac{\delta}{2}\right)^3 - \left(t-\tfrac{\delta}{2}\right)^3\right] = \frac{4\pi}{3}\left[3\delta t^2 + \tfrac{1}{4}\delta^3\right]. \tag{8}$$

Perturbing one point of M after the other, we can bound the error by (8) each time. Using the intensity, $\rho = \#M/\mathrm{Vol}[U]$, this implies

$$\left|\psi_k^A(t) - \psi_k^Q(t)\right| \leq \rho\Delta \leq \rho\frac{4\pi}{3}\left[3\delta t^2 + \tfrac{1}{4}\delta^3\right]. \tag{9}$$

We can eliminate the dependence on t by observing that for each k there is a value of t beyond which the k-th density functions of A and Q vanish. To determine this value, consider a point $y \in \mathbb{R}^3$ and the sets $A \cap B(y;t)$ and $Q \cap B(y;t)$. By the definition of R, the balls of radius R centered at the points of A cover $B(y;t-R)$, and similarly for Q. It follows that the two sets contain at least $(t/R-1)^3$ points each. Setting $k+1 \leq (t/R-1)^3$, we see that for $t \geq R\sqrt[3]{k+1} + R$, both sets have at least $k+1$ points each. Equivalently, y is covered by at least $k+1$ balls of radius t. Since this holds for every point $y \in \mathbb{R}^3$, we have $\psi_k^A(t) = \psi_k^Q(t) = 0$ for all $t \geq R\sqrt[3]{k+1} + R$. Note that $R\sqrt[3]{k+1} + R \leq 2R\sqrt[3]{k+1}$ for all $k \geq 0$. Replacing t in (9) by the latter bound, we get

$$\frac{1}{\sqrt[3]{k+1}^2}\|\psi_k^A - \psi_k^Q\|_\infty \leq 16\pi\rho R^2\delta + \frac{\pi}{3}\rho\delta^3 \leq \frac{12R^2}{r^3}\delta + \frac{1}{4r^3}\delta^3, \tag{10}$$

in which we use $\rho\frac{4\pi}{3}r^3 \leq 1$ to get the final inequality. Using $\delta^2 < r^2 < R^2$, this gives $C = 13R^2/r^3$ as an upper bound for the Lipschitz constant. ◀

Figure 3 illustrates Theorem 4.2 for a periodic set, A, and its perturbation, Q, in $\mathbb{R}^2$ by showing the first eight (undamped) density functions for both sets in different colors.

5 Completeness

The fingerprint map is *complete* if it is injective up to isometries; that is: non-isometric periodic sets are mapped to different fingerprints. We prove completeness generically, i.e. on a dense open subset; compare to [4, 18, 12]. The density fingerprint also distinguishes non-generic sets for which other means fail, as will be illustrated by an example in Section 5.2. The completeness of the fingerprint for all periodic sets, however, remains an open question. Indeed, at the time of writing this paper, the authors are not aware of a 3-dimensional

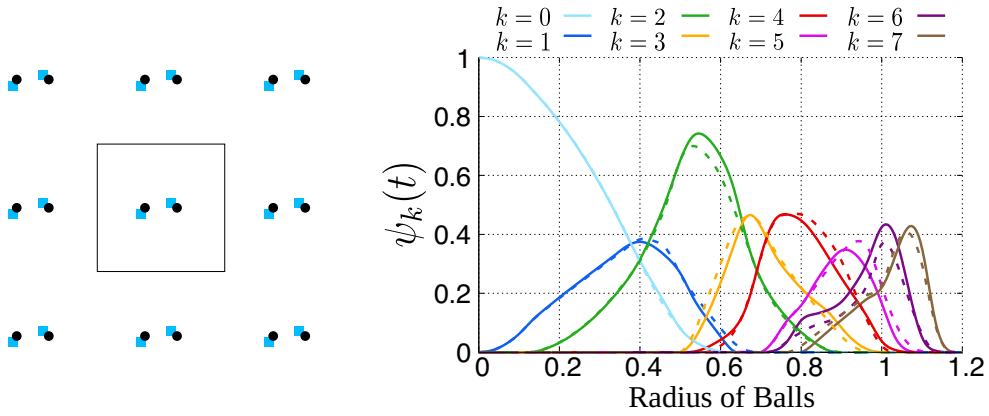


Figure 3 *Left*: a periodic set with two *black* points in its square unit cell, and the perturbed periodic set with two *blue* points in the same unit cell. *Right*: the graphs of the density functions are *solid* for the original set and *dashed* for the perturbed set. As predicted by Theorem 4.2, the small perturbation of the periodic set causes a small change in the fingerprint.

counterexample to completeness, but there is a 1-dimensional counterexample due to Morteza Saghafian: letting $U = \{0, 4, 9\}$ and $V = \{0, 1, 3\}$, it can be checked that the finite sets $U + V$ and $U - V$, and the periodic sets $15\mathbb{Z} + (U + V)$ and $15\mathbb{Z} + (U - V)$ cannot be distinguished by the 1-dimensional density fingerprint map.

5.1 Generic Completeness

We prove the completeness of the density fingerprint map for generic periodic sets in $\mathbb{R}^3$. The notion of genericity is defined by conditions that are satisfied by an open and dense subset of the space of periodic point sets. We formulate such conditions in terms of the *circumradius* of edges, triangles, and tetrahedra, which is the radius of the smallest sphere that passes through the vertices of the simplex. To avoid infinitely many constraints, we introduce an upper bound on the circumradii to consider. Denoting by $L = L(A, \vartheta)$ the list of all edges (pairs), triangles (triples) and tetrahedra (quadruples) spanned by points of a periodic set A whose circumradius is at most ϑ, we call A *generic for a constant threshold,* ϑ, if – apart from necessary violations due to periodicity – it satisfies the following three conditions:

I. the circumradii of different simplices in L are different;
II. the circumradii of different edges in L are not related to each other by a factor of 2.
III. for every $t \leq \vartheta$, there is at most one set of six circumradii of simplices in L such that the edges with twice their lengths assemble to a tetrahedron whose circumradius is t.

We call an edge a *lattice edge* if its length is the distance between two lattice points. Lattice edges violate Condition II and can thus be identified as such. A *lattice triangle* has three lattice edges, and a *lattice tetrahedron* has six lattice edges. The important difference between lattice and non-lattice simplices is that only the latter are unique up to lattice translations.

Since Conditions I, II, III can be phrased via finitely many algebraic equations in the vectors $x \in M, v_1, v_2, v_3$, the set of generic periodic sets with threshold ϑ is is open and dense in the space of all periodic sets with at most m motif points. We write $\mathrm{Rad}(A)$ for the largest finite circumradius of $p \leq 4$ points in A with pairwise distance at most four times the diameter of the unit cell. Since the diameter is the distance between two lattice points, this implies that $\mathrm{Rad}(A)$ is at least double the diameter.

▶ **Theorem 5.1** (Generic Completeness). *Let $A, Q \subseteq \mathbb{R}^3$ be non-isometric periodic sets that are generic for the threshold $\vartheta = \max\{\mathrm{Rad}(A), \mathrm{Rad}(Q)\}$. Then $\Psi(A) \neq \Psi(Q)$.*

Proof. Let $[A]$ denote the isometry class of A. We prove the unique reconstruction of $[A]$ from $\Psi(A)$ in two steps:

$$\Psi(A) \xrightarrow{\text{Step 1}} \text{tetrahedra in } L(A,\vartheta), \text{ up to isometries} \xrightarrow{\text{Step 2}} [A]$$

Step 1: Each density function is a weighted sum of the volumes of intersections of 2, 3, or 4 balls around points of A; see [9, Equation (5)]. The volume formulas of such intersections are given in [8]. It is cumbersome but not difficult to prove that they are piecewise analytic, and that the circumradii of edges, triangles and tetrahedra spanned by points of A are the positions where the functions are not analytic. Therefore, the set of all positions up to ϑ where at least one density function is not analytic yields the set of circumradii of simplices of L. We avoid the technicalities of using the differences between the left- and right-derivatives to distinguish which of these are caused by 2, 3, or 4 balls meeting, with the following trick. We treat all circumradii as if they were circumradii of edges, multiply them by two (to get the edge length), and try to assemble six of these edge lengths to form a tetrahedron. Whenever this gives a circumradius of a simplex of L, we have found a tetrahedron of A by Condition III. This way we can uniquely construct all tetrahedra of L up to isometries.

Step 2: To start the process, we choose a non-lattice tetrahedron from the list. If there is no such tetrahedron, then A is a lattice and can be reconstructed from the lexicographically shortest lattice tetrahedron from the list – i.e. the tetrahedron consisting of the shortest lattice edge, the second-shortest lattice edge (linearly independent from the first), and so on – defining a (Minkowski-)reduced [14] and therefore primitive unit cell of A. On the other hand, if there exist non-lattice tetrahedra, we choose the lexicographically shortest one, $abcd$, with non-lattice edge ab.

Placing $abcd$ in space – as we are only interested in the isometry class of A, we can place it arbitrarily – we identify all tetrahedra $abce$ from the list that have abc as a face and try to glue them onto $abcd$. There are two possibilities (related by a reflection) of how to glue $abce$; we denote the two different tip positions by e_1 and e_2. We prove that at most one of the two options gives a positive result when checking if the tetrahedron $abde_i$ is in the list of tetrahedra from Step 1: The triangles abd and abe are non-lattice, and therefore unique in A up to lattice translations by Condition I. Thus, when glued along ab, they span a uniquely defined tetrahedron $abde$ with a certain edge length de that is the distance between d and e_i for at most one of its two possible positions.

This gluing procedure yields (among others) all points of distance at most four times the diameter of the unit cell to a, b, c, d (by definition of $\mathrm{Rad}(A)$), except the ones that lie on a plane spanned by the triangles abc or abd. This neighborhood is large enough such that it contains every motif point at least once and such that it contains a lattice basis, which can be identified by computing the pairwise differences between the reconstructed points and checking whether they satisfy Condition II. Repeating the reconstructed points with respect to the lattice yields the isometry class of A. As the construction was unique given the genericity conditions, we get $\Psi(A) \neq \Psi(Q)$. ◀

5.2 Distinguishing Non-Generic Periodic Sets

There are indications that the density fingerprint map distinguishes all periodic sets and not just the generic ones. We now give the reason for our optimism. Example 5.2 describes two periodic sets that violate the above genericity conditions and can nevertheless be distinguished

by the density fingerprint map. On the other hand, the two sets can neither be distinguished by their density nor by their X-ray diffraction patterns; two means commonly used in crystallography to determine the structure of a crystal. X-ray diffraction patterns give all pairwise distance vectors of the periodic set, but they do not determine the isometry class of a periodic set [16]: there exist *homometric structures*, which are non-isometric periodic sets with the same 2-point autocorrelation functions; that is: identical multisets of pairs, up to translation. There even exist periodic sets with the same 2- and 3-point autocorrelation functions, as we now explain.

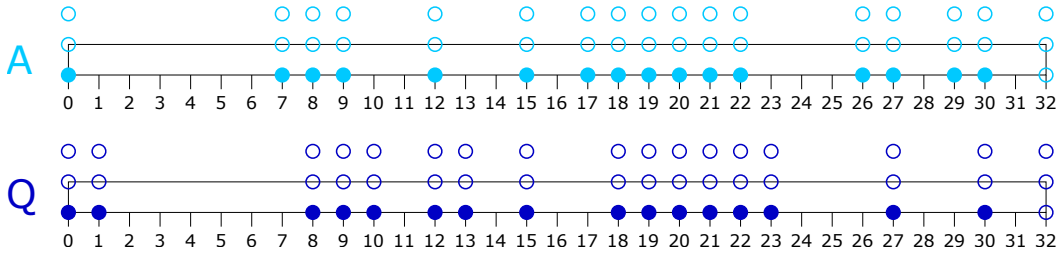


Figure 4 Periodic sets A and Q from Example 5.2, pictured with rectangular unit cells in two dimensions, for simplicity. Filled dots belong to the motifs while unfilled dots show the periodicity.

▶ **Example 5.2.** Let $A^{(1)}$ and $Q^{(1)}$ be sets with periodicity 32 in $\mathbb{R}$, each with 16 points in the corresponding motif:

$$0, 7, 8, 9, 12, 15, 17, 18, 19, 20, 21, 22, 26, 27, 29, 30; \tag{11}$$

$$0, 1, 8, 9, 10, 12, 13, 15, 18, 19, 20, 21, 22, 23, 27, 30; \tag{12}$$

see Figure 4. The authors of [11, Section 5.3] show that $A^{(1)}$ and $Q^{(1)}$ have the same 2- and 3-point autocorrelation functions. Taking the Cartesian product with $\mathbb{Z}^2$ preserves the equality between the autocorrelation functions, which yields periodic sets, $A, Q \subseteq \mathbb{R}^3$, with matching 2- and 3-point autocorrelation functions. Nevertheless, our density fingerprint map distinguishes them, as shown in Table 1: the L_∞-distances between the first four corresponding density functions vanish but the next five L_∞-distances are strictly positive.

Table 1 L_∞-distances between the corresponding density functions of the sets A and Q in Example 5.2.

k	0	1	2	3	4	5	6	7	8
$\lVert\psi_k^A - \psi_k^Q\rVert_\infty$	0.000	0.000	0.000	0.000	0.005	0.007	0.013	0.022	0.007

6 Computation

The algorithm for the density fingerprint map is based on two related geometric concepts: the k-th Dirichlet–Voronoi domain and the k-Brillouin zone of a point. After introducing both, we explain how they are used, and how much time it takes to construct them.

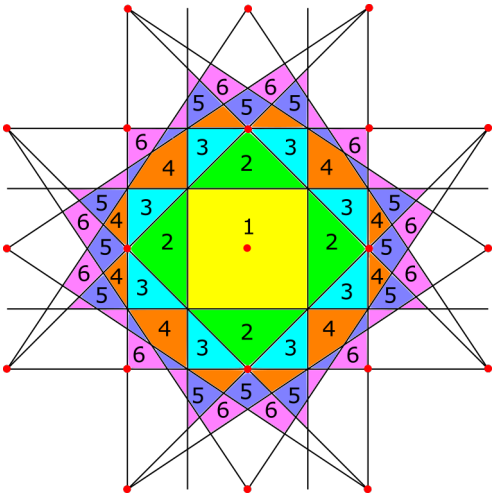


Figure 5 The 6-th Dirichlet–Voronoi domain of the point in the center decomposed into the first six Brillouin zones, which are indicated by colors and labels.

6.1 Dirichlet–Voronoi Domains and Brillouin Zones

Let $A \subseteq \mathbb{R}^3$ be a locally finite set of points. For every positive integer k, the *k-th Dirichlet–Voronoi domain* of a point $a \in A$ is the set of points in $\mathbb{R}^3$ for which a is among the k closest points in A, and the *k-th Brillouin zone* is the difference between the k-th and the $(k-1)$-st Dirichlet–Voronoi domains:

$$\mathrm{dom}_k(a, A) = \{x \in \mathbb{R}^3 \mid \|x - b\| < \|x - a\| \text{ for at most } k-1 \text{ points } b \in A\}, \tag{13}$$

$$\mathrm{zone}_k(a, A) = \mathrm{dom}_k(a, A) \setminus \mathrm{dom}_{k-1}(a, A); \tag{14}$$

see Figure 5. Here we set $\mathrm{dom}_0(a, A) = \emptyset$ so that the first Brillouin zone is well defined. Note that $\mathrm{zone}_k(a, A)$ is the set of points $x \in \mathbb{R}^3$ for which there are exactly $k-1$ points $b \in A$ that are closer to x than a is. Observe also that $\mathrm{dom}_k(a, A)$ is closed and star-convex, and if A is Delone, then it is also compact. If A is a lattice, $A = \Lambda$, then all k-th Dirichlet–Voronoi domains are translates of each other, and similarly for the Brillouin zones: $\mathrm{dom}_k(a, \Lambda) = \mathrm{dom}_k(0, \Lambda) + a$ and $\mathrm{zone}_k(a, \Lambda) = \mathrm{zone}_k(0, \Lambda) + a$. Except for a measure zero subset of $\mathbb{R}^3$, every point x has a unique k-closest point in Λ. This implies that the k-th Brillouin zones *tile* $\mathbb{R}^3$, by which we mean that their closures cover $\mathbb{R}^3$ while their interiors are pairwise disjoint. These properties generalize to a periodic set, $A = M + \Lambda$: the k-th Brillouin zones of the points in $a + \Lambda$ are translates of each other, and the k-th Brillouin zones of all $a \in A$ tile $\mathbb{R}^3$.

6.2 Decomposed Multiple Cover

Assume from here on that $A = M + \Lambda$ is a periodic set. To compute φ_k^A, we may use any fundamental domain of the lattice. Particularly convenient is the union of the k-th Brillouin zones of the points in M as it lends itself to finding the subset covered by at least k of the balls.

▶ **Theorem 6.1** (Density for Periodic Set). *Let $A = M + \Lambda$ be periodic with lattice $\Lambda \subseteq \mathbb{R}^3$ and motif $M \subseteq U$ in the unit cell of Λ, and let $k \geq 1$ be an integer. Then the probability that a random point $x \in U$ belongs to k or more balls of radius $t \geq 0$ centered at the points of A is*

$$\varphi_k^A(t) = \tfrac{1}{\mathrm{Vol}[U]} \sum_{a \in M} \mathrm{Vol}[\mathrm{zone}_k(a, A) \cap B(a; t)]. \tag{15}$$

Proof. Let M_k be the union of the k-th Brillouin zones of the points $a \in M$ and note that $M_k + \Lambda$ tiles $\mathbb{R}^3$. It follows that $\mathrm{Vol}[M_k] = \mathrm{Vol}[U]$. Let $x \in M_k$ be in the interior of $\mathrm{zone}_k(a, A)$. By construction, a is the unique k-closest point to x, so x lies in k or more balls if and only if $x \in B(a; t)$. Summing over all points $a \in M$ gives (15). ◀

Clearly, $\varphi_0^A(t) = 1$ for all radii t. Given $k \geq 0$ and $t \geq 0$, we use (15) to compute $\varphi_k^A(t)$ and $\varphi_{k+1}^A(t)$, and we get $\psi_k^A(t) = \varphi_k^A(t) - \varphi_{k+1}^A(t)$. To implement (15), we need to compute the volume of the intersection of a ball with a convex polyhedron. We could, for example, decompose the polyhedron into tetrahedra and use explicit expressions for the volume of intersections between balls and simplices; see for example [3]. A C++ implementation evaluating the density functions for a given periodic set using this strategy can be found at [19]. Alternatively, we could use inclusion-exclusion, which allows for further consolidation of the formula, writing $\varphi_k^A(t)$ as an alternating sum of common intersections of up to four balls each. This does not lead to any asymptotic improvements of the running time, so we omit further details and refer to [9] instead.

6.3 Algorithm and Running Time

To evaluate the density functions $\psi_0^A, \psi_1^A, \ldots, \psi_k^A$ at a value t, we compute a plane arrangement for each point $a \in M$ that consists of enough planes so that the first $k+1$ Brillouin zones of a occur. Specifically, for a large enough radius, s, we consider for each $b \neq a$ in $A \cap B(a; s)$ the *bisector* of b and a, which is the plane defined by $\|x - a\| = \|x - b\|$. These bisectors decompose $\mathbb{R}^3$ into convex cells. We refer to this decomposition as the *arrangement* of the planes. The 3-dimensional cells that are separated from a by exactly $j-1$ planes form the *j-th belt* of the arrangement.

We now address the question how small we can choose s such that the first $k+1$ belts are the first $k+1$ Brillouin zones of a. To begin, we recall that $t \geq 2R\sqrt[3]{k+1}$ implies that $\psi_i(t) = 0$ for $0 \leq i \leq k$; see the proof of Theorem 4.2. To express this insight geometrically, let $R_{k+1}(a)$ be the maximum distance of a point in the $(k+1)$-st Brillouin zone of a from a. That the density functions ψ_0^A to ψ_k^A are zero for $t \geq 2R\sqrt[3]{k+1}$ implies $\varphi_{k+1}^A(t) = 1$, for these values of t, and therefore $R_{k+1}(a) \leq 2R\sqrt[3]{k+1}$. To capture all the relevant planes, it thus suffices to consider all points $b \in A \setminus \{a\}$ at distance at most $s = 2R_{k+1}(a)$ from a. Using a straightforward volume argument, we see that $B(a; 2R_{k+1}(a))$ contains at most $(4R\sqrt[3]{k+1} + r)^3/r^3 = \mathcal{O}(k)$ points, in which we treat r and R as constants.

Constructing the arrangement of $\mathcal{O}(k)$ planes incrementally, as described in [7, chapter 7], takes time $\mathcal{O}(k^3)$. Doing this for each point in the motif takes time $\mathcal{O}(\#M \cdot k^3)$, and within the same time bound we can evaluate the first $k+1$ density functions.

7 An Application to Crystal Structure Prediction

Crystal Structure Prediction (CSP) aims to predict whether a selected molecule can exist as a functional material, i.e. a crystal with useful functions or properties. In other words, CSP seeks to answer the question of whether copies of a molecule can be arranged in such a way that the resulting crystal is stable (will not deform and lose its properties over time) as well as useful. Crucially, CSP tries to answer this question without setting foot in a laboratory, with the hope of dramatically reducing the need to perform the time-consuming process of physically synthesizing crystals.

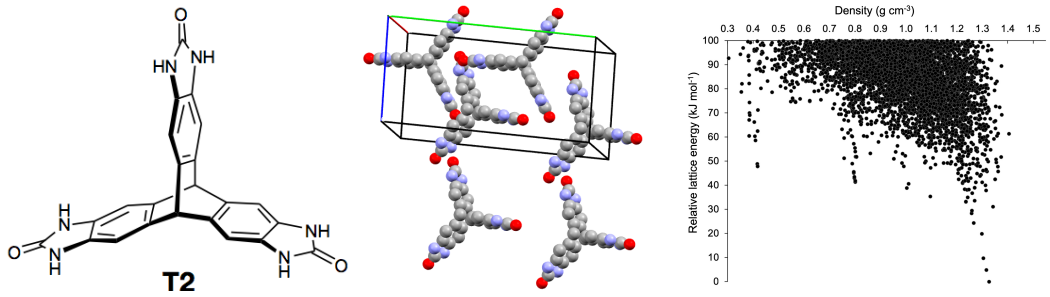


Figure 6 *Left*: a T2 molecule. *Middle*: the T2-δ crystal with highlighted unit cell. *Right*: the output of CSP for the T2 molecule. It is a plot of 5679 simulated T2 crystal structures [17, Fig. 2d], each represented by two coordinates: the physical density (atomic mass within a unit cell divided by the unit cell volume) and energy (determining the crystal's thermodynamic stability). Structures at the bottom of the "downward spikes" are likely to be stable.

Our collaborators at Liverpool's Materials Innovation Factory [17] used CSP to predict that the T2 molecule (Figure 6) can be crystallized into a new structure that has half the physical density of the only previously known structure for T2, a desirable property for applications such as gas storage. As part of this process, they also identified four other structures of interest. Following the CSP predictions, they synthesized 5 families of T2-crystals in the laboratory by varying parameters like temperature and pressure, calling them T2-α, T2-β, ..., T2-ϵ. One of them, T2-γ, indeed had the desired property of having only half the physical density of the previously known structure T2-α. They scanned the synthesized crystals using X-ray powder diffraction yielding Crystallographic Information Files, each containing the unit cell and the motif points representing the atoms. These files were then compared with the results of the simulations, either by using their physical density alongside the COMPACK algorithm – which compares only a finite portion of the structure – or by looking at visualizations of the crystal structures. This comparison showed that the synthesized crystals matched the prediction well. Our collaborators deposited these structures into the globally used Cambridge Structural Database.

At a later time, we used our newly developed fingerprints to verify our collaborators' matchings between the synthesized crystals T2-α to T2-ϵ and the simulated crystals entry 99, 28, 62, 09, 01. We did so by computing, for each of the five matches, the distance between the density functions of the synthesized and the simulated crystal. As one is the prediction of the other, we expected to see small distances. And for four of the five structures this was true: T2-γ, for example, always has an L_∞-distance of less than 0.04 over the first eight pairs of corresponding density functions; see Table 2. However, when we came to check the distances between density functions of T2-δ with its predicted structure, we were surprised to see large distances (the final row of Table 2). It turned out that a mix-up of files had happened, and what was uploaded to the Cambridge Structural Database as T2-δ was in fact T2-β' (a crystal from the T2-β family). The density fingerprint revealed this error, which was verified by chemists upon a visual inspection, and it is because of this that T2-δ was subsequently correctly deposited.

Plots of the density functions of correctly matched synthesized and simulated structures can be seen in Figure 7. As another application, we expect that the fingerprint will be used to simplify the large output data sets produced by CSP by comparing simulated structures with each other, thus speeding up what is currently a slow process.

Table 2 *First five rows:* the L_∞-distances between the first eight pairs of corresponding density functions of physically synthesized T2 crystals (T2-α, T2-β, etc.) and the simulated structures that had predicted them from the CSP output dataset (entry XX). *Last row:* the suspiciously larger numbers revealed the mix-up of the files T2-δ and T2-β' and thus led to depositing the initially omitted Crystallographic Information File of the T2-δ crystal into the Cambridge Structural Database.

$\|\psi_k^A - \psi_k^Q\|_\infty$	$k=0$	1	2	3	4	5	6	7
T2-α vs entry 99	0.0042	0.0092	0.0125	0.0056	0.0099	0.0088	0.0127	0.0099
T2-β vs entry 28	0.0157	0.0156	0.0159	0.0224	0.0334	0.0396	0.0357	0.0454
T2-γ vs entry 62	0.0020	0.0080	0.0128	0.0155	0.0153	0.0250	0.0296	0.0391
T2-δ vs entry 09	0.0610	0.0884	0.1267	0.0676	0.0915	0.0801	0.0733	0.0388
T2-ϵ vs entry 01	0.0132	0.0152	0.0207	0.0571	0.0514	0.0431	0.0468	0.0550
T2-β' vs entry 09	0.2981	0.2631	0.3718	0.3747	0.2563	0.2360	0.3161	0.3232

8 Discussion

The main contribution of this paper is a fingerprint map from periodic sets in $\mathbb{R}^3$ (which model crystals) to series of density functions. This map is obviously invariant under isometries, and we prove it is continuous and generically complete. We leave the completeness without genericity assumption as an open question. In this context, it is worth noticing that our proof of generic completeness makes only limited use of the order, k, at which the circumradius of an edge, triangle, or tetrahedron is detected. Recall that the order is the number of points in the respective circumsphere. Is this additional information sufficient to prove general completeness?

A drawback of the bottleneck distance between periodic sets used in this paper is its sensitivity to changes of the unit cell; see Lemma 4.1. An alternative dissimilarity that may be more relevant in practice considers affine transformations, τ, that minimize the bottleneck distance:

$$d_{\mathrm{AT}}(A,Q) = \inf_{\tau} \max\{\min\{d_B(A,\tau(Q)), d_B(\tau(A),Q)\}, |\log s_1|, |\log s_3|\}, \tag{16}$$

in which $s_1 \geq s_2 \geq s_3$ are the three singular values of the matrix of τ. Is the density fingerprint map defined in Section 3 continuous with respect to this dissimilarity?

We close this paper with three extensions of the results presented in this paper. Different types of atoms are often modeled as balls with different radii. A possible geometric formalism is that of weighted points and the power distance [2]. Our geometric results generalize to this setting, but some need a careful adaptation. Our continuity result for periodic sets (Theorem 4.2) also generalizes to non-periodic Delone sets that allow for a reasonable definition of density functions. Considering that quasiperiodic crystals can be modeled as such, it might be worthwhile to find out how far such an extension can be pushed. Finally, we mention that our results generalize to arbitrary finite dimension.

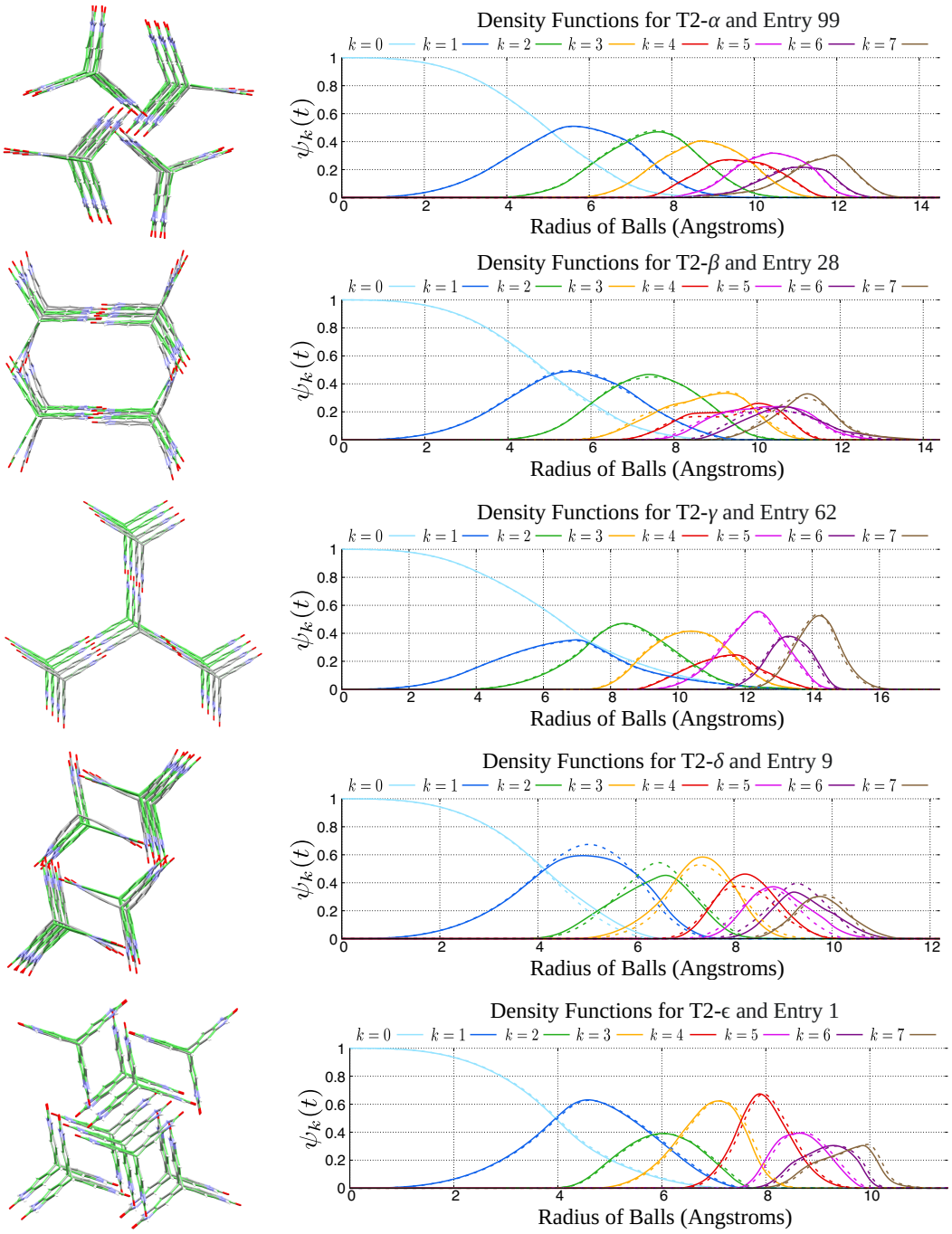


Figure 7 *Left*: experimental T2 crystals (curved gray molecules) and their simulated versions (straight green molecules) overlaid. *Right*: the density functions of the periodic sets of molecular centres of the experimental T2 crystals (solid curves) vs. simulated crystals (dashed curves).

References

1 Lawrence C. Andrews, Herbert J. Bernstein, and G.A. Pelletier. A perturbation stable cell comparison technique. *Acta Crystallographica*, A36(2):248–252, 1980. doi:10.1107/S0567739480000496.

2 Franz Aurenhammer. Power diagrams: properties, algorithms and applications. *SIAM Journal on Computing*, 16(1):78–96, 1987. doi:10.1137/0216006.

3 David Avis, Binay K. Bhattacharya, and Hiroshi Imai. Computing the volume of the union of spheres. *The Visual Computer*, 3(6):323–328, 1988. doi:10.1007/BF01901190.

4 Mireille Boutin and Gregor Kemper. On reconstructing n-point configurations from the distribution of distances or areas. *Advances in Applied Mathematics*, 32(4):709–735, 2004. doi:10.1016/S0196-8858(03)00101-5.

5 James A. Chisholm and Sam Motherwell. Compack: a program for identifying crystal structure similarity using distances. *Journal of Applied Crystallography*, 38(1):228–231, 2005. doi:10.1107/S0021889804027074.

6 Nikolai P. Dolbilin, Jeffrey C. Lagarias, and Marjorie Senechal. Multiregular point systems. *Discrete & Computational Geometry*, 20(4):477–498, 1998. doi:10.1007/PL00009397.

7 Herbert Edelsbrunner. *Algorithms in Combinatorial Geometry*, volume 10 of *Monographs in Theoretical Computer Science. An EATCS Series.* Springer-Verlag Berlin Heidelberg, 1987. doi:10.1007/978-3-642-61568-9.

8 Herbert Edelsbrunner and Ping Fu. Measuring space filling diagrams and voids. In *Molecular Biophysic Report UIUC-BI-MB-94–01*. University of Illinois at Urbana-Champaign, 1994.

9 Herbert Edelsbrunner and Mabel Iglesias-Ham. Multiple covers with balls I: inclusion–exclusion. *Computational Geometry*, 68:119–133, 2018. doi:10.1016/j.comgeo.2017.06.014.

10 Gemma de la Flor, Danel Orobengoa, Emre Tasci, Juan M. Perez-Mato, and Mois I. Aroyo. Comparison of structures applying the tools available at the Bilbao crystallographic server. *Journal of Applied Crystallography*, 49(2):653–664, 2016. doi:10.1107/S1600576716002569.

11 F. Alberto Grünbaum and Calvin C. Moore. The use of higher-order invariants in the determination of generalized patterson cyclotomic sets. *Acta Crystallographica*, A51(3):310–323, 1995. doi:10.1107/S0108767394009827.

12 Paul Lemke, Steven S. Skiena, and Warren D. Smith. *Reconstructing sets from interpoint distances.* Springer Berlin Heidelberg, 2003. doi:10.1007/978-3-642-55566-4_27.

13 Clare F. Macrae, Paul R. Edgington, Patrick McCabe, Elna Pidcock, Greg P. Shields, Robin Taylor, Matthew Towler, and Jacco van de Streek. Mercury: visualization and analysis of crystal structures. *Journal of Applied Crystallography*, 39(3):453–457, 2006. doi:10.1107/S002188980600731X.

14 Phong Q. Nguyen and Damien Stehlé. Low-dimensional lattice basis reduction revisited. *In "Proceedings of the 6th International Algorithmic Number Theory Symposium"*, pages 338–357, 2004. doi:10.1007/978-3-540-24847-7_26.

15 Paul Niggli. *Krystallographische und strukturtheoretische Grundbegriffe*, volume 1 of *Handbuch der Experimentalphysik, Band 7.* Akademische Verlagsgesellschaft mbH, 1928.

16 Linus Pauling and Maple D. Shappell. 8. The crystal structure of bixbyite and the c-modification of the sesquioxides. *Zeitschrift für Kristallographie – Crystalline Materials*, 75(1):128–142, 1930. doi:10.1515/zkri-1930-0109.

17 Angeles Pulido et al. Functional materials discovery using energy–structure–function maps. *Nature*, 543(7647):657–664, 2017. doi:10.1038/nature21419.

18 Marjorie Senechal. A point set puzzle revisited. *European Journal of Combinatorics*, 29(8):1933–1944, 2008. doi:10.1016/j.ejc.2008.01.013.

19 Philip Smith. Density functions of a periodic set in C++, 2020. URL: https://github.com/Phil-Smith1/Density_Functions.

On the Edge Crossings of the Greedy Spanner

David Eppstein
Department of Computer Science, University of California, Irvine, CA, USA

Hadi Khodabandeh
Department of Computer Science, University of California, Irvine, CA, USA

Abstract

The greedy t-spanner of a set of points in the plane is an undirected graph constructed by considering pairs of points in order by distance, and connecting a pair by an edge when there does not already exist a path connecting that pair with length at most t times the Euclidean distance. We prove that, for any $t > 1$, these graphs have at most a linear number of crossings, and more strongly that the intersection graph of edges in a greedy t-spanner has bounded degeneracy. As a consequence, we prove a separator theorem for greedy spanners: any k-vertex subgraph of a greedy spanner can be partitioned into sub-subgraphs of size a constant fraction smaller, by the removal of $O(\sqrt{k})$ vertices. A recursive separator hierarchy for these graphs can be constructed from their planarizations in linear time, or in near-linear time if the planarization is unknown.

2012 ACM Subject Classification Theory of computation → Sparsification and spanners; Theory of computation → Computational geometry; Theory of computation → Design and analysis of algorithms

Keywords and phrases Geometric Spanners, Greedy Spanners, Separators, Crossing Graph, Sparsity

Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.33

Related Version *Full Version*: `https://arxiv.org/abs/2002.05854`

Funding This work was supported in part by the US National Science Foundation under grant CCF-1616248.

1 Introduction

Geometric spanners are geometric graphs whose distances approximate distances in complete graphs, while having fewer edges than complete graphs. Given a set of points V on the Euclidean plane (or in any other metric space), a t-spanner on V can be defined as a graph S having V as its set of vertices V and satisfying the following inequality for every pair of points (P, Q):

$$d_S(P, Q) \leq t \cdot d(P, Q) \tag{1}$$

where $d_S(P, Q)$ is the length of the shortest path between P and Q using the edges in S, and $d(P, Q)$ is the Euclidean distance of P and Q. We call Equation 1 the *bounded stretch property*. Because of this inequality, t-spanners provide a t-approximation for the pairwise distances between the set of points in V. The parameter $t > 1$ is called the *stretch factor* or *spanning ratio* of the spanner and determines how accurate the approximate distances are; spanners having smaller stretch factors are more accurate.

Spanners can be defined in any metric space, but they are often located in a geometric space, where a heavy or undesirable network is given and finding a sparse and light-weight spanner and working with it instead of the actual network makes the computation easier and faster. Finding light-weight geometric spanners has been a topic of interest in many areas of computer science, including communication network design and distributed computing. These subgraphs have few edges and are easy to construct, leading them to appear in a wide range of applications since they were introduced [14, 38, 45]. In wireless ad hoc networks t-spanners

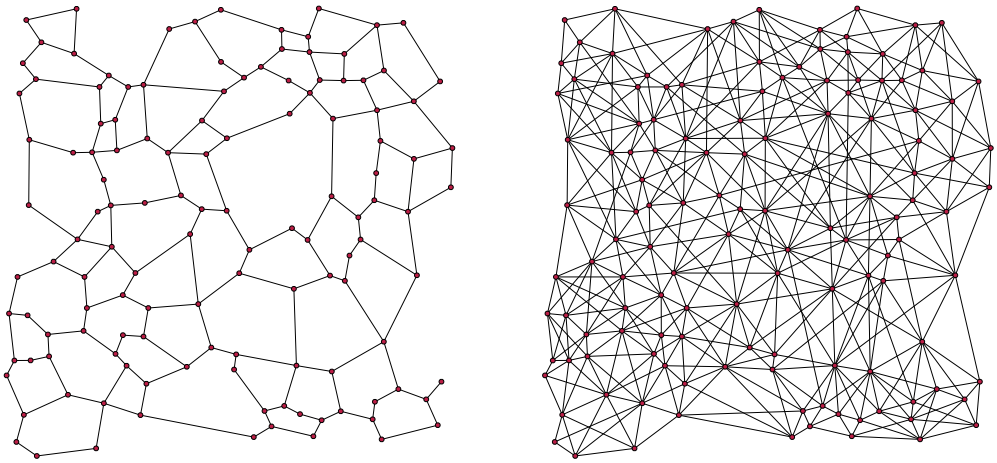


Figure 1 Greedy spanners of 128 random points with stretch factor 2 (left) and 1.1 (right).

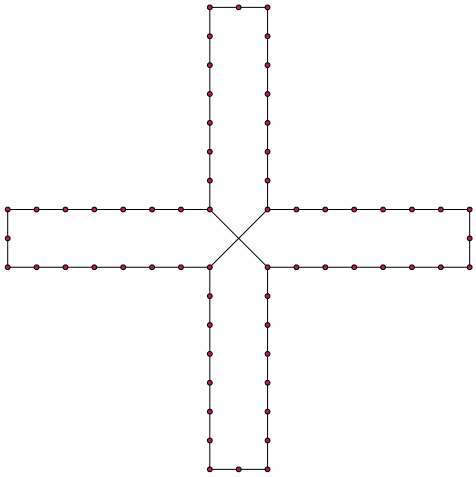


Figure 2 Nonplanar greedy spanner with stretch factor 11.3.

are used to design sparse networks with guaranteed connectivity and guaranteed bounds on routing length [5]. In distributed computing spanners provide communication-efficiency and time-efficiency through the sparsity and the bounded stretch property [9, 23, 7, 24]. There has also been extensive use of geometric spanners in the analysis of road networks [25, 2, 13]. In robotics, geometric spanners helped motion planners to design near-optimal plans on a sparse and light subgraph of the actual network [19, 43, 16]. Spanners have many other applications including computing almost shortest paths [22, 15, 46, 30], and overlay networks [12, 47, 37].

Researchers have developed various construction techniques for spanners, depending on the specific additional properties needed in these applications. Well-separated pair decomposition, θ-graphs, and greedy spanners are among the most well-known of these geometric spanner constructions. Here, we focus on the greedy spanner. It was first introduced by Althöfer [3, 4] and Bern, generalizing a pruning strategy used by Das and Joseph [17] on a triangulation of the planar graph [25].

A greedy spanner can be constructed by running the greedy spanner algorithm (Algorithm 1) on a set of points on the Euclidean plane. This short procedure adds edges one at a time to the spanner it constructs, in ascending order by length. For each pair of vertices, in this order, it checks whether that pair already satisfies the bounded stretch inequality using the edges already added. If not, it adds a new edge connecting the pair. Therefore, by construction, each pair of vertices satisfies the inequality, either through previous edges or (if not) through the newly added edge. The resulting graph is therefore a t-spanner. Examples of the results of this algorithm, for two different stretch factors, are shown in Figure 1. Although the 2-spanner in the figure is planar, this is not true for 2-spanners in general: there exist point sets with non-planar greedy t-spanners for arbitrarily large values of t (Figure 2), and by placing widely-spaced copies of the same construction within a single point set, one can construct point sets whose greedy t-spanners have linearly many crossings, for arbitrarily large values of t.

Algorithm 1 The naive greedy spanner algorithm.

```
1: procedure NAIVE-GREEDY(V)
2:     Let S be a graph with vertices V and edges E = {}
3:     for each pair (P, Q) ∈ V^2 in increasing order of d(P, Q) do
4:         if d_S(P, Q) > t · d(P, Q) then
5:             Add edge PQ to E
       return S
```

A naïve implementation of the greedy spanner algorithm runs in time $\mathcal{O}(n^3 \log n)$, where n is the number of given points [11]. Bose et al. [11] improved the running time of Algorithm 1 to near-quadratic time using a bounded version of Dijkstra's algorithm. Narasimhan et al. proposed an approximate version of the greedy spanner algorithm that reached a running time of $\mathcal{O}(n \log n)$, based on the use of approximate shortest path queries [18, 36, 44].

Despite the simplicity of Algorithm 1, Farshi and Gudmundsson [29] observed that in practice, greedy spanners are surprisingly good in terms of the number of edges, weight, maximum vertex degree, and also the number of edge crossings. Many of these properties have been proven rigorously. Filster and Solomon [31] proved that greedy spanners have size and lightness that is optimal to within a constant factor for worst-case instances. They also achieved a near-optimality result for greedy spanners in spaces of bounded doubling dimension. Borradaile, Le, and Wulff-Nilsen [10] recently proved optimality for doubling metrics, generalizing a result of Narasimhan and Smid [44], and resolving an open question posed by Gottlieb [35], and Le and Solomon showed that no geometric t-spanner can do asymptotically better than the greedy spanner in terms of number of edges and lightness [41]. However, past work has not proven rigorous bounds on the number of crossings of greedy spanners.

One reason for particular interest in bounds on the number of crossings is the close relation, for geometric graphs in the plane, between crossings and *separators*. The well-known planar separator theorem of Lipton and Tarjan [42] states that any planar graph (that is, a geometric graph with no crossings) can be partitioned into subgraphs whose size is at most a constant fraction of the total by the removal of $O(\sqrt{n})$ vertices. This property is central to the efficiency of many algorithms on planar graphs [34, 21, 27, 26, 40], and applied as well in multiple computational geometry problems [32, 6, 39]. Analogous separator theorems have been extended from planar graphs to graphs with few crossings per edge [20], or more generally to graphs with sparse patterns of crossings [28, 8]. Past work has not shown that greedy spanners have small separators, but as we will show, bounds on their crossings can be used to show that they do.

1.1 Our Contribution

In this paper we prove that greedy t-spanners in the Euclidean plane have few crossings, for any $t > 1$, and we use this result (together with a result of Eppstein and Gupta [28] on graphs with sparse patterns of crossings) to prove that greedy spanners in the Euclidean plane have small separators. In particular, we prove:

- **Claim 1.** Each edge in a greedy spanner can be crossed by only $O(1)$ edges of equal or greater length, where the constant in the $O(1)$ depends only on t, the stretch factor of the spanner. More precisely as $t \to 1$ there are $O(1/(t-1)^2)$ edges that cross the given edge and are longer than it by a factor of $\Omega(1/(t-1))$ (Theorem 14), and $1/(t-1)^{O(1)}$ edges that cross the given edge and have length at least ϵ times it, for any constant $\epsilon > 0$ (Theorem 17).
- **Claim 2.** For some choices of t, there exist greedy spanners in which some edges are crossed by a linear number of (significantly shorter) edges.
- **Claim 3.** Every n-vertex greedy spanner, and every n-vertex subgraph of a greedy spanner, can be partitioned into connected components of size at most cn for a constant $c < 1$ by the removal of $O(\sqrt{n})$ vertices. Again, the constant factor in the $O(\sqrt{n})$ term depends only on the stretch factor of the spanner. Moreover, a separator hierarchy for the greedy spanner can be constructed from its planarization in near-linear time (Theorem 20).

It is known that the spanners that are constructed by some other methods, i.e. semi-separated pair decomposition [1] and hierarchical decomposition [33], have small $\mathcal{O}(\sqrt{n})$-separators in two dimensions. Although experimental results of Farshi and Gudmundsson on greedy spanners of random point sets had shown the number of crossings to be small in practice [29] our results are the first theoretical results on this property, the first to study crossings for worst-case and not just random instances, and the first to prove that greedy spanners have small $\mathcal{O}(\sqrt{n})$-separators.

1.2 Intuition

Our proof that edges can be crossed by only a bounded number of edges of greater or equal length splits into two cases, one for crossings by edges of significantly greater length and another for crossings by edges of similar length.

For edges of significantly greater length, we divide the greedy spanner edges that might cross the given edge into a constant number of nearly-parallel sets of edges, and prove the bound separately within each such set. We show that, within a set of nearly-parallel long edges that all cross the given edge, the edges can be totally ordered by their projections onto a base line, because edges whose endpoints project to nested intervals would contradict the greedy property of the spanner (the inner of two nested edges could be used to shortcut the outer one). By similar reasoning, the endpoints of any two nearly-parallel long crossing edges are separated by a distance that is at least a constant fraction of the length of the smaller edge. This geometric growth in the separation of the endpoints leads to a system of inequalities on the lengths of the edges that can only be satisfied when the number of crossing edges is bounded by a constant.

For edges of comparable length to the crossed edge, we use a grid to partition the crossing edges into a constant number of subsets of edges, such that within each subset all edges have pairs of endpoints that are close to each other relative to the length of the edge, and we show that each of these subsets can contain only a unique edge.

Our construction showing that a single edge can be crossed linearly many times is based on the combination of three "zig-zag" sets of points, evenly spaced in their x-coordinates and alternating between two different y-coordinates. In the top and bottom zig-zag, the distance along the zigzag between two consecutive points with the same y-coordinates is exactly t times the difference between their x-coordinates, while in the middle zig-zag it is slightly greater. The greedy spanner for this point set contains the zig-zag edges, plus a single long edge crossing all of the middle edges, for a pair of points that are far enough from each other along the middle zig-zag for their Euclidean distance to be almost the same as their difference in x-coordinates (differing by a number smaller than the amount by which a single edge of the middle zig-zag exceeds t times its difference in x-coordinates).

The results on separators follow from previous results on the existence of separators in graphs whose edge intersection graphs have bounded degeneracy [28].

2 Preliminaries

As we mentioned earlier, t-spanners can be defined in any metric space. For a given graph G, a t-spanner is defined in the following way,

▶ **Definition 1** (t-spanner). *Given a metric graph $G = (V, E, d)$, i.e. weighted graph with distances as weights, a t-spanner is a spanning subgraph S of G such that for any pair of vertices $u, w \in V$,*

$$d_G(u, w) \leq t \cdot d(u, w)$$

where $d_G(u, w)$ is the length of the shortest path in G between u and w.

Then the greedy spanner on a given set of points V can be defined in the following way,

▶ **Definition 2** (greedy spanner). *Given a set of points V in any metric space, a greedy spanner on V is a t-spanner that is an output of Algorithm 1.*

Here we restrict the problem to geometric graphs and we take advantage of inequalities that hold in geometric space.

We consider the natural embedding that the greedy spanner inherits from its vertices. Edges are drawn as straight segments between the two points corresponding to the two endpoints of the edge. We say two edges of the spanner cross or intersect if their corresponding segments intersect at some interior point. The crossing graph of a given embedding can be defined in this way,

▶ **Definition 3** (crossing graph). *Given a graph $G(V, E)$ and its Euclidean embedding, the crossing graph $Cr(G)$ is a graph $G'(E, C)$ whose vertices are the edges of the original graph and for each two vertices $e, f \in E$ there is an edge between them if and only if they intersect with each other in the embedding given for G.*

Most of the proofs here use a lemma that we call the *short-cutting lemma*, which is simple but very useful in greedy spanners. The lemma is proven in [44] and it states that a t-spanner edge cannot be shortcut by some other edges of the spanner by a factor of t. Formally,

▶ **Lemma 4** (short-cutting). *An edge AB of a greedy t-spanner cannot be shortcut by some other spanner edges by a factor of t, i.e. there is no constant k and points $A = P_0, P_1, \ldots, P_k = B$ that $P_0P_1, P_1P_2, \ldots, P_{k-1}P_k$ are all spanner edges distinct from AB, and*

$$\sum_{i=0}^{k-1} |P_iP_{i+1}| \leq t \cdot |AB|.$$

If some of the segments P_iP_{i+1} are not included in the spanner, the same argument still works but a factor t appears before the term $|P_iP_{i+1}|$ in the summation. So

▶ **Corollary 5** (Extended short-cutting). *Given a greedy t-spanner S and an edge AB of S, there cannot be a constant k and points $A = P_0, P_1, \ldots, P_k = B$ such that*

$$\sum_{P_iP_{i+1} \in S} |P_iP_{i+1}| + t \cdot \sum_{P_iP_{i+1} \notin S} |P_iP_{i+1}| \leq t \cdot |AB|.$$

The proof of Lemma 4 and Corollary 5 are included in the full version for reference. In the following section we consider intersections between an arbitrary edge of a greedy spanner and sufficiently larger edges, and we show a constant bound on the number of intersections per edge. In Section 3.5 we again prove a constant bound for the number of intersections between a spanner edge and other edges of almost the same length. Finally, we introduce an example in which the number of intersections with smaller edges can be more than any constant bound, completing our analysis. In Section 4 we introduce some new results and improvements based on the constant bound we provided earlier.

3 Few intersections with long edges

In this section, we prove an upper bound on the number of intersections of an edge with sufficiently larger edges. We will specifically show that the number of intersections, in this case, has a constant bound that only depends on t. Later in Section 3.5 we prove a constant bound also exists for the intersections with the edges that have almost the same length of the intersecting edge. Hence we prove our first claim.

In this setting, we consider an arbitrary edge AB of the spanner, and we are interested in counting the number of intersections that AB may have with sufficiently larger edges, i.e. edges PQ that intersect AB at some interior point with $|PQ| > c \cdot |AB|$ for some constant $c > 1$ which we will specify later.

First, we only consider a set of *almost-parallel* spanner segments that cross AB, where we define the term *almost-parallel* below, and we put a bound on the number of these segments. Then we generalize the bound to hold for all large spanner segments that cross AB.

3.1 Definitions

▶ **Definition 6** (almost-parallel). *We say a pair of arbitrary segments PQ and RS in the plane are* almost-parallel *or θ-*parallel *if there is an angle of at most θ between them. We say a set of segments are* almost-parallel *if every pair of segments chosen from the set are almost-parallel.*

For any set of almost-parallel segments, we define a baseline to measure the angles and distances with respect to that line.

▶ **Definition 7** (baseline). *Given a set of almost-parallel (or θ-parallel) segments in the plane, denoted by S, the* baseline *$b(S)$ of the set of segments S is the segment with the smallest slope.*

We use the uniqueness of the segment chosen in Definition 7 and we emphasize that any other definition works if it determines a unique segment for any almost-parallel set of segments.

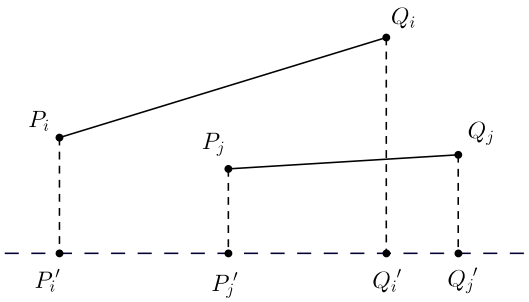


Figure 3 Ordering segments by projecting on the baseline l, here $P_iQ_i <_{\mathcal{R}} P_jQ_j$.

In Section 3.2, we define a total ordering on a set of almost-parallel segments that cross a spanner segment AB. Once we have sorted these segments based on the ordering, in Section 3.3 we prove the distance between the endpoints of two consecutive segments is at least a constant fraction of the length of the smaller segment. Putting together these two parts, in Section 3.4 we prove there cannot be more than a constant number of segments in the sequence.

3.2 A total ordering on almost-parallel intersecting segments

In this section, we define an ordering on a set of almost-parallel segments of the t-spanner. The ordering is based on the order of the projections of the endpoints of the segments on the baseline corresponding to the segments. We first define the ordering and then we use Lemma 9 and Lemma 10 to prove that it is a total ordering when the set of almost-parallel segments are all crossing a given segment of the spanner.

Consider a set of almost-parallel spanner segments that cross some spanner segment. One can define an ordering on this set of almost-parallel segments, which we call the *endpoint-ordering*, based on how their endpoints are ordered along the direction they are aligned to. We formulate the definition in the following way,

▶ **Definition 8** (endpoint-ordering). *Let $S = \{P_iQ_i : i = 1, 2, \ldots, k\}$ be a set of almost-parallel segments. Also let l be the baseline of S, $b(S)$. Define the* endpoint-ordering $\mathcal{R}$ *between two segments P_iQ_i and P_jQ_j by projecting the endpoints P_i, P_j, Q_i, Q_j to the baseline l and comparing the order of the projected points P_i', P_j', Q_i', Q_j' along an arbitrary direction of the baseline l,*

- *$P_iQ_i <_{\mathcal{R}} P_jQ_j$ if the projections are ordered as $P_i'P_j'Q_i'Q_j'$ or $P_i'Q_i'P_j'Q_j'$.*
- *$P_iQ_i >_R P_jQ_j$ if they are ordered as $P_j'P_i'Q_j'Q_i'$ or $P_j'Q_j'P_i'Q_i'$. (Figure 3)*

We claim that the endpoint-ordering is a total ordering on the set of almost-parallel segments. This basically means that after projecting two almost-parallel segments on the baseline, none of the resulting projections would lie completely inside the other one. Other cases correspond to a valid endpoint-ordering.

In order to prove this, first, we prove a simpler case when the two segments intersect with each other. This assumption will help to significantly simplify the proof. Later we use this lemma to show the original claim is also true.

▶ **Lemma 9.** *Let MN and PQ be two intersecting segments from a set of θ-parallel spanner segments. Also assume that $\theta < \frac{t-1}{2t}$ where t is the stretch factor of the spanner. Then MN and PQ are endpoint-ordered, i.e. the projection of one of the segments on the baseline of the set cannot be included in the projection of the other one.*

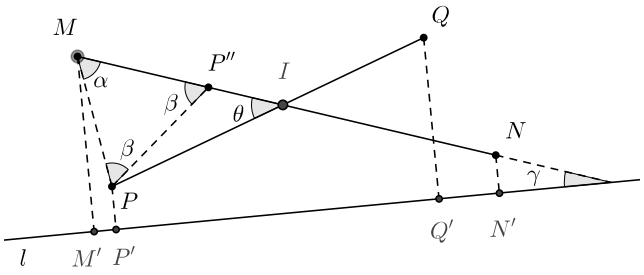


Figure 4 Proof of Lemma 9.

Proof. We prove the lemma by contradiction. Without loss of generality suppose that the projections of P and Q on some baseline l are both between the projections of M and N (on the same baseline). We show that MN can be shortcut by PQ by a factor of t, i.e.

$$t \cdot |MP| + |PQ| + t \cdot |QN| \leq t \cdot |MN|.$$

Let P', Q', M', and N' be the corresponding projections of P, Q, M, and N on l, respectively (Figure 4). Also let I be the intersection point and $\alpha = \angle PMI$, and also γ to be the angle between MN and the baseline, according to the figure. By the assumption P' is between M' and N', so $\alpha \leq \pi/2 + \gamma \leq \pi/2 + \theta$. Let also P'' be the point on MN s.t. $|MP''| = |MP|$ and $\beta = \angle MPP'' = \angle MP''P$. Then by sine law,

$$\frac{|MI| - |MP|}{|PI|} = \frac{|P''I|}{|PI|} = \frac{\sin(\beta - \theta)}{\sin\beta} = \frac{\sin(\pi/2 - \alpha/2 - \theta)}{\sin(\pi/2 - \alpha/2)} = \frac{\cos(\alpha/2 + \theta)}{\cos(\alpha/2)}$$
$$= \cos\theta - \sin\theta\tan(\alpha/2) \tag{2}$$

but we have,

$$\cos\theta \geq 1 - \theta^2/2 \geq 1 - \theta/4 \tag{3}$$

as $\theta < \frac{t-1}{2t} < 1/2$. Also,

$$\tan(\alpha/2) \leq \tan(\pi/4 + \theta/2) = \tan(\pi/4 + 1/4) < \frac{7}{4}. \tag{4}$$

Putting together Equation 2, Equation 3, and Equation 4, also using $\sin\theta \leq \theta$,

$$\frac{|MI| - |MP|}{|PI|} \geq (1 - \theta/4) - (\frac{7}{4})\theta = 1 - 2\theta > \frac{1}{t}$$

which is equivalent to $t \cdot |MI| - t \cdot |MP| \geq |PI|$. Similarly, $t \cdot |NI| - t \cdot |NQ| \geq |QI|$. Adding together,

$$t \cdot |MN| - t \cdot |MP| - t \cdot |NQ| \geq |PQ|$$

which is what we are looking for. ◀

Lemma 9 assumes that segments intersect at some interior point. In order to prove the totality of the ordering, we also need to prove the claim when the segments do not intersect with each other. Instead, in this case, both segments intersect some spanner edge. We use Lemma 9 to prove this in the Lemma 10.

▶ **Lemma 10.** *Let MN and PQ be two segments chosen from a set of θ-parallel spanner segments that cross a spanner edge AB. Also assume that $\theta < \frac{t-1}{2(t+1)}$, and $\min(|MN|, |PQ|) \geq \frac{3t(t+1)}{t-1}|AB|$, where t is the spanner parameter. Then MN and PQ are endpoint-ordered.*

The proof of this lemma is included in the full version due to space limits. Based on Lemma 10 it is easy to prove the main result of this section, Proposition 11.

▶ **Proposition 11.** *Given an arbitrary edge AB of a t-spanner, for a set of sufficiently large almost-parallel spanner edges that intersect AB, the* endpoint-ordering *we defined in Definition 8 is a total ordering.*

Proof. Totality requires reflexivity, anti-symmetry, transitivity, and comparability. Reflexivity and transitivity are trivial because of the projection. Anti-symmetry and comparability follow directly from Lemma 10. ◀

Now that we have ordered the set of almost-parallel spanner segments, we can prove a lower bound on the distance of two ordered segments. Later we prove a bound on the number of these segments based on the resulting distance lower bound.

3.3 Lower bounding the distance of endpoints of two crossing segments

In Section 3.2 we restricted the problem to a set of almost-parallel spanner segments that intersect another spanner segment, and we defined an ordering on these segments. The next step is to find a lower bound on the distance of two almost-parallel segments that intersect some spanner segment AB. The idea is to show that both endpoints of two ordered segments cannot be arbitrarily close, and hence there cannot be more than a constant number of them in a sequence.

More specifically, we show in Proposition 13 that the corresponding endpoints of two almost-parallel spanner segments that both cross the same spanner segment should have a distance of at least a constant fraction of the length of the smaller segment, otherwise the longer segment could be shortcut by the smaller one, which is indeed a contradiction. A weaker version of this lemma is proven in [44] and it is called "gap property", but the inequality we show here is actually stronger.

First we propose a geometric inequality in Lemma 12 that helps to prove the proposition. Then we complete the proof of the proposition at the end of this section.

▶ **Lemma 12.** *Let MN and PQ be two segments in the plane with angle θ. Then*

$$||MN| - |PQ|| > ||MP| - |NQ|| - 2\sin(\theta/2) \cdot |PQ|.$$

Proof. By swapping MN and PQ, it turns out that the case where $|MN| \geq |PQ|$ is stronger than $|MN| \leq |PQ|$. So without loss of generality, let $|MN| \geq |PQ|$ and by symmetry $|MP| \geq |NQ|$. Let Q' be the rotation of Q around P by θ, so that PQ' and MN are parallel, and $|PQ'| = |PQ|$ (Figure 5). Let Q'' be the point on the ray PQ' where $|PQ''| = |MN|$. As a result Q'' and P will be on different sides of Q'. By the triangle inequality,

$$\begin{aligned} |MN| - |PQ| &= |PQ''| - |PQ'| = |Q'Q''| \geq |NQ''| - |NQ'| \\ &= |MP| - |NQ'| \geq |MP| - (|NQ| + |QQ'|) \\ &= |MP| - |NQ| - 2|PQ| \cdot \sin(\theta/2). \end{aligned}$$ ◀

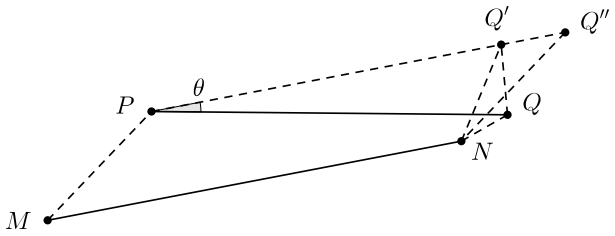


Figure 5 Proof of Lemma 12.

Now we state and prove Proposition 13. As we mentioned earlier, the idea is to show one of the segments can be shortcut by the other one if one of the matching endpoints is very close. In the simplest case when the segments are two opposite sides of a rectangle, it is easy to see that a distance of $\frac{t-1}{2}|PQ|$ on both sides is required to prevent short-cutting. In the general case, when the segments are placed arbitrarily, Proposition 13 holds.

▶ **Proposition 13.** *Let MN and PQ be two θ-parallel spanner segments. The matching endpoints of these two segments cannot be closer than a constant fraction of the length of the smaller segment. More specifically,*

$$\min(|MP|, |NQ|) \geq \frac{t - 1 - 2\sin(\theta/2)}{2t} \min(|MN|, |PQ|).$$

Proof. Without loss of generality and by symmetry, let $|NQ| \leq |MP|$. Suppose, on the contrary, that $|NQ| < \frac{t-1-2\sin(\theta/2)}{2t}|PQ|$. Then,

$$\begin{aligned} t \cdot |MP| + |PQ| + t \cdot |NQ| &\leq t \cdot (|MN| - |PQ| + |NQ| + 2\sin(\theta/2) \cdot |PQ|) + |PQ| + t \cdot |NQ| \\ &= t \cdot |MN| - (t - 1 - 2\sin(\theta/2))|PQ| + 2t \cdot |NQ| \\ &< t \cdot |MN|. \end{aligned}$$

So MN can be shortcut by PQ within a factor of t which contradicts the extended short-cutting lemma for the edge MN and the path $MPQN$. ◀

So far, in Proposition 13 we proposed an ordering on the set of almost-parallel spanner segments that cross a given edge and we proved each of these segments has a significant distance from the other ones. In the next section we put together these results and we find a constant upper bound on the number of these segments.

3.4 Putting together

Based on the ordering proposed in Section 3.2, and the lower bound we proved in Section 3.3, we can show that the following constant upper bound on the number of intersections with sufficiently large edges holds.

If we look at one of the endpoints of the endpoint-ordered sequence of almost-parallel spanner segments, and we project them on the baseline, the distance of every two consecutive projected points cannot be smaller than a constant fraction of the length of the smaller segment, i.e. $|P'_iP'_{i+1}| \geq C \cdot \min(|P_iQ_i|, |P_{i+1}Q_{i+1}|)$ for all values of $i = 0, 1, \ldots, k-1$. Summing up these inequalities leads to a bound on k, the number of segments.

▶ **Theorem 14.** *For sufficiently small θ, the number of sufficiently large θ-parallel segments that intersect a given edge AB of a t-spanner is limited by*

$$\frac{4t}{(t-1-2\sin(\theta/2))\cos\theta}+1.$$

By sufficiently large we specifically mean larger than $\frac{3t(t+1)}{t-1}|AB|$.

Proof. Let P_iQ_is be the segments larger than AB that intersect AB at some angle in $[\alpha,\alpha+\theta)$. Let P_0Q_0 be the shortest edge among P_iQ_is. Because of the total ordering, at least half of the segments are larger than P_0Q_0 with respect to the ordering $\mathcal{R}$, or at least half of them are smaller than P_0Q_0 with respect to $\mathcal{R}$. Without loss of generality, assume that half of the segments are larger than P_0Q_0 with respect to $\mathcal{R}$, and they are indexed by $i=1,2,\ldots,(k-1)/2$. Also let P_i's and Q_i's be the projections of P_is and Q_is on the base line l. By Proposition 13, for all i, P_{i+1} is farther than P_i by a constant fraction of $\min(|P_iQ_i|,|P_{i+1}Q_{i+1}|)$, so

$$\begin{aligned}\sum_{i=0}^{(k-3)/2}|P_iP_{i+1}| &> \frac{t-1-2\sin(\theta/2)}{2t}\sum_{i=0}^{(k-3)/2}\min(|P_iQ_i|,|P_{i+1}Q_{i+1}|)\\ &\geq \frac{t-1-2\sin(\theta/2)}{2t}\cdot\frac{k-1}{2}|P_0Q_0|.\end{aligned}$$

If $k\geq\frac{4t}{(t-1-2\sin(\theta/2))\cos\theta}+1$,

$$\sum_{i=0}^{(k-3)/2}|P_iP_{i+1}|>\frac{1}{\cos\theta}|P_0Q_0|$$

or equivalently

$$|P_0Q_0|<\sum_{i=0}^{(k-3)/2}|P_iP_{i+1}|\cos\theta\leq\sum_{i=0}^{(k-3)/2}|P_i'P_{i+1}'|=|P_0'P_{\frac{k-1}{2}}'|,$$

which is not possible, because $P_0'P_{\frac{k-1}{2}}'$ lies inside $P_0'Q_0'$ and so $|P_0'P_{\frac{k-1}{2}}'|\leq|P_0'Q_0'|\leq|P_0Q_0|$ which contradicts the last inequality above. ◀

The constraints on θ imposed by our earlier lemmas imply that, as $t\to1$, we should choose θ proportional to $t-1$. Asymptotically, as $t\to1$, the number of large segments of all angles that intersect AB is $O(1/(t-1)^2)$, with one factor of $1/(t-1)$ coming from the bound in the theorem and the second factor coming from the number of different classes of nearly-parallel segments.

3.5 Almost-equal length edges

In the previous subsections, we proved a bound on the number of intersections with relatively larger edges. Here we prove a constant bound on the number of intersections with edges that are nearly the same length as the length of the intersecting edge. In the full version we also consider intersections with relatively smaller edges, which completes our analysis for this problem.

For same-length intersections Lemma 10 does not hold anymore, hence the endpoint-ordering is not necessarily a total ordering in this case. Since totality is a key requirement for the rest of the proof the same proof will not work anymore. But Proposition 13 still holds as it has no assumption on the ordering of the segments.

Our idea is to partition the neighborhood of AB into a square network, such that no two spanner segments can have both endpoints in the same squares (Figure 6). If this happens, then by Proposition 13 one of the segments should be shortcut by the other one, leading to a contradiction because both segments are already included in the spanner.

We first prove a simpler version of Proposition 13 that does not include θ in the inequality, as we are not using the almost-parallel assumption and the value of θ can be large enough to make the inequality in Proposition 13 trivial. We will use this modified version to prove our claim.

▶ **Lemma 15.** *Given a greedy spanner with parameter t and two spanner segments MN and PQ,*

$$\max(|MP|, |NQ|) \geq \frac{t-1}{2t} \min(|MN|, |PQ|).$$

Proof. Suppose on the contrary that

$$\max(|MP|, |NQ|) < \frac{t-1}{2t} \min(|MN|, |PQ|).$$

Also, without loss of generality assume that $|MN| \geq |PQ|$. Then,

$$t \cdot |MP| + |PQ| + t \cdot |NQ| \leq (t-1) \min(|MN|, |PQ|) + |PQ| = t \cdot |PQ| \leq t \cdot |MN|$$

which contradicts the extended short-cutting lemma for the edge MN and the path $MPQN$. ◀

▶ **Proposition 16.** *The number of spanner segments PQ that cross a segment AB of a t-spanner and that have length within $\alpha \cdot |AB| \leq |PQ| \leq \beta \cdot |AB|$ is limited by*

$$\left\lceil \frac{2\beta(2\beta+1)}{\alpha^2} \cdot \frac{8t^2}{(t-1)^2} \right\rceil^2$$

where t is the spanner parameter.

Proof. Partition the area around AB with squares of edge length $\frac{t-1}{2\sqrt{2}t} \cdot \alpha|AB|$ with edges parallel or perpendicular to AB. The area that an endpoint of a crossing segment can lie in is a rectangle of size $(2\beta+1)|AB|$ by $2\beta|AB|$ (Figure 6). The total number of squares in this area would be

$$\frac{2\beta(2\beta+1)}{\alpha^2} \cdot \frac{8t^2}{(t-1)^2}.$$

But for each crossing segment the pair of squares that contain the two endpoints of the segment is unique. Otherwise two segments, e.g. MN and PQ, will have both endpoints at the same pair, which means

$$\max(|MP|, |NQ|) < (\sqrt{2})(\frac{t-1}{2\sqrt{2}t} \cdot \alpha|AB|) = \frac{t-1}{2t} \cdot \alpha|AB| \leq \frac{t-1}{2t} \min(|MN|, |PQ|)$$

which cannot happen due to Lemma 15. So the total number of pairs, and hence the total number of crossing segments, would be

$$\left\lceil \frac{2\beta(2\beta+1)}{\alpha^2} \cdot \frac{8t^2}{(t-1)^2} \right\rceil^2.$$ ◀

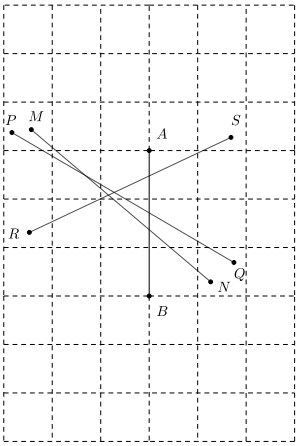


Figure 6 Partition of the area around AB.

In Proposition 16 both α and β can be chosen arbitrarily, and the bound is a strictly increasing function of β and a strictly decreasing function of α. The bound tends to infinity when β is large enough, and also when α is small enough. So it basically does not prove any constant bound for the cases that edges are very small or very large. But for the edges of almost the same length, it gives a constant upper bound.

Putting together the main results of Section 3 and Section 3.5 we can prove the following bound for the number of intersections with not-relatively-small spanner segments.

▶ **Theorem 17.** *Given a spanner segment AB in the Euclidean plane and a positive constant ϵ, the number of edges of length at least $\epsilon \cdot |AB|$ of the spanner that intersect AB is $\mathcal{O}(\frac{t^{12}}{\epsilon^4(t-1)^8})$.*

Proof. By Theorem 14 the number of intersections with edges PQ such that $|PQ| \geq \frac{3t(t+1)}{t-1}|AB|$ is bounded by

$$C_1 = \frac{4t}{(t-1-2\sin(\theta/2))\cos\theta} + 1 \in \mathcal{O}(\frac{t}{t-1}).$$

On the other side, putting $\alpha = \epsilon$ and $\beta = \frac{3t(t+1)}{t-1}$ into Proposition 16 implies that the number of intersections with edges larger than AB and smaller than $\frac{3t(t+1)}{t-1}|AB|$ is at most

$$C_2 = \left[\frac{2}{\epsilon^2}\left(\frac{3t(t+1)}{t-1}\right)\left(2\frac{3t(t+1)}{t-1}+1\right)\left(\frac{8t^2}{(t-1)^2}\right)\right]^2 \in \mathcal{O}(\frac{t^{12}}{\epsilon^4(t-1)^8}).$$

Hence the number of intersections with edges larger than AB is at most $C_1 + C_2$, which is $\mathcal{O}(\frac{t^{12}}{\epsilon^4(t-1)^8})$. ◀

In Section 3 we proved the number of intersections with sufficiently large edges is bounded by a constant and now we completed the proof for all larger edges. In the full version of the paper, we also show that the same argument does not work for intersections with arbitrarily smaller edges, and we provide an example that shows there can be an arbitrarily large number of intersections with smaller edges. This completes our analysis of the problem. In the following section, we show some of the applications of this result, most importantly, the sparsity of the crossing graph of the greedy spanner.

4 Separators

In this section, we use the crossing bound that we proved in Theorem 17 to show that greedy spanners have small separators. First, we start with the definition of *degeneracy*, which is a measure of sparsity of a graph.

▶ **Definition 18** (degeneracy). *A graph G is called k-degenerate, if each subgraph of G has a vertex of degree at most k. The smallest value of k for which a graph is k-degenerate is called the* degeneracy *of the graph.*

The first important consequence of Theorem 17 is the constant degeneracy of the crossing graph of the greedy spanner, implying its sparsity and linearity of the number of edges, i.e. crossing.

▶ **Theorem 19.** *The crossing graph of a greedy t-spanner has a constant degeneracy.*

Proof. In any subgraph of the crossing graph, by Theorem 17 the node corresponding to the smallest edge has at most a constant number of neighbours. ◀

This, together with the result of [28] implies the existence of sublinear separators for greedy spanners. A separator is a subset of vertices whose removal splits the graph into smaller pieces. A sublinear separator is a sublinear number of vertices with the same property. The splitting can be recursively performed on the smaller parts and a separator hierarchy can be constructed in this way, which effectively helps in the design of new recursive algorithms. The planarization of a graph, which is obtained by adding new vertices on the edge intersections of the graph, would help us to find such hierarchy in linear time, otherwise, a near-linear time algorithm would be used.

▶ **Theorem 20.** *Greedy spanners have separators of size $\mathcal{O}(\sqrt{n})$. Also, a separator hierarchy for them can be constructed from their planarization in linear time.*

Proof. By Theorem 19 the crossing graph of the greedy t-spanner has a constant degeneracy, so by Theorem 6.9 of [28] they have separators of size $\mathcal{O}(\sqrt{n})$. Also by the same theorem, a separator hierarchy for them can be constructed from their planarization in linear time. ◀

One of the basic algorithms that can be improved using the separator hierarchy is Dijkstra's single-source shortest path algorithm, which runs in $\mathcal{O}(n \log n)$ time on a graph with n vertices. As a result of Theorem 20 linear algorithms exist for finding single-source shortest path on greedy spanners, If the planarization has not already been found, it can be constructed in time $O(n \log^{(i)} n)$ for any constant i, where $\log^{(i)}$ denotes the i-times iterated logarithm, e.g. $\log^{(3)} n = \log \log \log n$ [27].

▶ **Corollary 21.** *Single source shortest paths can be computed in time $O(n \log^{(i)} n)$ on a greedy spanner.*

Proof. This follows from the planarization algorithm and from the existence and construction of separators from planarizations by Corollary 6.10 of [28]. ◀

5 Conclusions

We have shown that greedy t-spanners in the plane have linearly many crossings, and that the intersection graphs of their edges have bounded degeneracy but can have unbounded (and even linear) degree. As a consequence, we proved that these graphs have small separators.

Given these results, it is natural to ask whether higher-dimensional Euclidean greedy t-spanners also have small separators. This cannot be achieved through bounds on crossings, because in dimensions greater than two, graphs whose vertices are in general position can have no crossings. We leave this question open for future research.

References

1. Mohammad A Abam and Sariel Har-Peled. New constructions of sspds and their applications. *Computational Geometry*, 45(5-6):200–214, 2012.
2. Mohammad Ali Abam, Mark De Berg, Mohammad Farshi, and Joachim Gudmundsson. Region-fault tolerant geometric spanners. *Discrete & Computational Geometry*, 41(4):556–582, 2009. doi:10.1007/s00454-009-9137-7.
3. Ingo Althöfer, Gautam Das, David Dobkin, and Deborah Joseph. Generating sparse spanners for weighted graphs. In John R. Gilbert and Rolf Karlsson, editors, *Proceedings of the 2nd Scandinavian Workshop on Algorithm Theory (SWAT)*, volume 447 of *Lecture Notes in Computer Science*, pages 26–37. Springer, 1990. doi:10.1007/3-540-52846-6_75.
4. Ingo Althöfer, Gautam Das, David Dobkin, Deborah Joseph, and José Soares. On sparse spanners of weighted graphs. *Discrete & Computational Geometry*, 9(1):81–100, 1993. doi:10.1007/BF02189308.
5. Khaled Alzoubi, Xiang-Yang Li, Yu Wang, Peng-Jun Wan, and Ophir Frieder. Geometric spanners for wireless ad hoc networks. *IEEE Transactions on Parallel and Distributed Systems*, 14(4):408–421, 2003. doi:10.1109/TPDS.2003.1195412.
6. Srinivasa Arikati, Danny Z. Chen, L. Paul Chew, Gautam Das, Michiel Smid, and Christos D. Zaroliagis. Planar spanners and approximate shortest path queries among obstacles in the plane. In Josep Diaz and Maria Serna, editors, *Proceedings of the 4th European Symposium on Algorithms (ESA)*, volume 1136 of *Lecture Notes in Computer Science*, pages 514–528. Springer, 1996. doi:10.1007/3-540-61680-2_79.
7. Baruch Awerbuch, Bonnie Berger, Lenore Cowen, and David Peleg. Near-linear time construction of sparse neighborhood covers. *SIAM Journal on Computing*, 28(1):263–277, 1998. doi:10.1137/S0097539794271898.
8. Sang Won Bae, Jean-Francois Baffier, Jinhee Chun, Peter Eades, Kord Eickmeyer, Luca Grilli, Seok-Hee Hong, Matias Korman, Fabrizio Montecchiani, Ignaz Rutter, and Csaba D. Tóth. Gap-planar graphs. *Theoretical Computer Science*, 745:36–52, 2018. doi:10.1016/j.tcs.2018.05.029.
9. Surender Baswana, Telikepalli Kavitha, Kurt Mehlhorn, and Seth Pettie. Additive spanners and (α, β)-spanners. *ACM Transactions on Algorithms*, 7(1):5, 2010. doi:10.1145/1868237.1868242.
10. Glencora Borradaile, Hung Le, and Christian Wulff-Nilsen. Greedy spanners are optimal in doubling metrics. In *Proceedings of the 30th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2371–2379. Society for Industrial and Applied Mathematics, 2019. doi:10.1137/1.9781611975482.145.
11. Prosenjit Bose, Paz Carmi, Mohammad Farshi, Anil Maheshwari, and Michiel Smid. Computing the greedy spanner in near-quadratic time. *Algorithmica*, 58(3):711–729, 2010. doi:10.1007/s00453-009-9293-4.
12. Rebecca Braynard, Dejan Kostic, Adolfo Rodriguez, Jeffrey Chase, and Amin Vahdat. Opus: an overlay peer utility service. In *Proceedings of the 5th IEEE Conference on Open Architectures and Network Programming (OPENARCH)*, pages 167–178. IEEE, 2002. doi:10.1109/OPNARC.2002.1019237.
13. Shiri Chechik, Michael Langberg, David Peleg, and Liam Roditty. Fault tolerant spanners for general graphs. *SIAM Journal on Computing*, 39(7):3403–3423, 2010. doi:10.1137/090758039.
14. Paul Chew. There are planar graphs almost as good as the complete graph. *Journal of Computer and System Sciences*, 39(2):205–219, 1989. doi:10.1016/0022-0000(89)90044-5.

15 Edith Cohen. Fast algorithms for constructing t-spanners and paths with stretch t. *SIAM Journal on Computing*, 28(1):210–236, 1998. doi:10.1137/S0097539794261295.

16 Gautam Das. The visibility graph contains a bounded-degree spanner. In *Proceedings of the 9th Canadian Conference on Computational Geometry (CCCG)*, pages 70–75, 1997. URL: https://cccg.ca/proceedings/1997/.

17 Gautam Das and Deborah Joseph. Which triangulations approximate the complete graph? In Hristo Djidjev, editor, *Proceedings of the International Symposium on Optimal Algorithms (OA)*, volume 401 of *Lecture Notes in Computer Science*, pages 168–192. Springer, 1989. doi:10.1007/3-540-51859-2_15.

18 Gautam Das and Giri Narasimhan. A fast algorithm for constructing sparse Euclidean spanners. *International Journal of Computational Geometry & Applications*, 7(04):297–315, 1997. doi:10.1142/S0218195997000193.

19 Andrew Dobson and Kostas E. Bekris. Sparse roadmap spanners for asymptotically near-optimal motion planning. *International Journal of Robotics Research*, 33(1):18–47, 2014. doi:10.1177/0278364913498292.

20 Vida Dujmović, David Eppstein, and David R. Wood. Structure of graphs with locally restricted crossings. *SIAM Journal on Discrete Mathematics*, 31(2):805–824, 2017. doi:10.1137/16M1062879.

21 Zdenek Dvorak and Sergey Norin. Strongly sublinear separators and polynomial expansion. *SIAM Journal on Discrete Mathematics*, 30(2):1095–1101, 2016. doi:10.1137/15M1017569.

22 Michael Elkin. Computing almost shortest paths. *ACM Transactions on Algorithms*, 1(2):283–323, 2005. doi:10.1145/1103963.1103968.

23 Michael Elkin and David Peleg. $(1+\varepsilon,\beta)$-spanner constructions for general graphs. *SIAM Journal on Computing*, 33(3):608–631, 2004. doi:10.1137/S0097539701393384.

24 Michael Elkin and Jian Zhang. Efficient algorithms for constructing $(1+\varepsilon,\beta)$-spanners in the distributed and streaming models. *Distributed Computing*, 18(5):375–385, 2006. doi:10.1007/s00446-005-0147-2.

25 David Eppstein. Spanning trees and spanners. In Jörg-Rüdiger Sack and Jorge Urrutia, editors, *Handbook of Computational Geometry*, pages 425–461. North-Holland, 2000. doi:10.1016/B978-044482537-7/50010-3.

26 David Eppstein and Michael T. Goodrich. Studying (non-planar) road networks through an algorithmic lens. In *Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages A16:1–A16:10. ACM, 2008. doi:10.1145/1463434.1463455.

27 David Eppstein, Michael T. Goodrich, and Darren Strash. Linear-time algorithms for geometric graphs with sublinearly many edge crossings. *SIAM Journal on Computing*, 39(8):3814–3829, 2010. doi:10.1137/090759112.

28 David Eppstein and Siddharth Gupta. Crossing patterns in nonplanar road networks. In *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages A40:1–A40:9. ACM, 2017. doi:10.1145/3139958.3139999.

29 Mohammad Farshi and Joachim Gudmundsson. Experimental study of geometric t-spanners. *ACM Journal of Experimental Algorithmics*, 14:1.3:1–1.3:29, 2009. doi:10.1145/1498698.1564499.

30 Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. Graph distances in the streaming model: the value of space. In *Proceedings of the 16th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 745–754. Society for Industrial and Applied Mathematics, 2005.

31 Arnold Filtser and Shay Solomon. The greedy spanner is existentially optimal. In *Proceedings of the 35th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC)*, pages 9–17. ACM, 2016. doi:10.1145/2933057.2933114.

32 Alan M. Frieze, Gary L. Miller, and Shang-Hua Teng. Separator based parallel divide and conquer in computational geometry. In *Proceedings of the 4th ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, volume 92, pages 420–429, 1992. doi:10.1145/140901.141934.

33 Martin Fürer and Shiva Prasad Kasiviswanathan. Spanners for geometric intersection graphs. In *Workshop on Algorithms and Data Structures*, pages 312–324. Springer, 2007.

34 Michael T. Goodrich. Planar separators and parallel polygon triangulation. *Journal of Computer and System Sciences*, 51(3):374–389, 1995. doi:10.1006/jcss.1995.1076.

35 Lee-Ad Gottlieb. A light metric spanner. In *Proceedings of the 56th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 759–772. IEEE, 2015. doi:10.1109/FOCS.2015.52.

36 Joachim Gudmundsson, Christos Levcopoulos, and Giri Narasimhan. Fast greedy algorithms for constructing sparse geometric spanners. *SIAM Journal on Computing*, 31(5):1479–1500, 2002. doi:10.1137/S0097539700382947.

37 Lujun Jia, Rajmohan Rajaraman, and Christian Scheideler. On local algorithms for topology control and routing in ad hoc networks. In *Proceedings of the 15th ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 220–229. ACM, 2003. doi:10.1145/777412.777447.

38 J. Mark Keil. Approximating the complete Euclidean graph. In Rolf Karlsson and Andrzej Lingas, editors, *Proceedings of the 1st Scandinavian Workshop on Algorithm Theory (SWAT)*, volume 318 of *Lecture Notes in Computer Science*, pages 208–213. Springer, 1988. doi:10.1007/3-540-19487-8_23.

39 David Kirkpatrick. Optimal search in planar subdivisions. *SIAM Journal on Computing*, 12(1):28–35, 1983. doi:10.1137/0212002.

40 Philip N. Klein, Shay Mozes, and Christian Sommer. Structured recursive separator decompositions for planar graphs in linear time. In *Proceedings of the 45th ACM Symposium on Theory of Computing (STOC)*, pages 505–514. ACM, 2013. doi:10.1145/2488608.2488672.

41 Hung Le and Shay Solomon. Truly optimal Euclidean spanners. In *Proceedings of the 60th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 1078–1100. IEEE, 2019. doi:10.1109/FOCS.2019.00069.

42 Richard J. Lipton and Robert Endre Tarjan. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics*, 36(2):177–189, 1979. doi:10.1137/0136016.

43 James D. Marble and Kostas E. Bekris. Asymptotically near-optimal planning with probabilistic roadmap spanners. *IEEE Transactions on Robotics*, 29(2):432–444, 2013. doi:10.1109/TRO.2012.2234312.

44 Giri Narasimhan and Michiel Smid. *Geometric Spanner Networks*. Cambridge University Press, 2007. doi:10.1017/CBO9780511546884.

45 David Peleg and Alejandro A. Schäffer. Graph spanners. *Journal of Graph Theory*, 13(1):99–116, 1989. doi:10.1002/jgt.3190130114.

46 Liam Roditty and Uri Zwick. On dynamic shortest paths problems. *Algorithmica*, 61(2):389–401, 2011. doi:10.1007/s00453-010-9401-5.

47 Wenjie Wang, Cheng Jin, and Sugih Jamin. Network overlay construction under limited end-to-end reachability. In *Proceedings of the 24th Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, volume 3, pages 2124–2134. IEEE, 2005.

On Ray Shooting for Triangles in 3-Space and Related Problems

Esther Ezra
School of Computer Science, Bar Ilan University, Ramat Gan, Israel

Micha Sharir
School of Computer Science, Tel Aviv University, Israel

Abstract

We consider several problems that involve lines in three dimensions, and present improved algorithms for solving them. The problems include (i) ray shooting amid triangles in $\mathbb{R}^3$, (ii) reporting intersections between query lines (segments, or rays) and input triangles, as well as approximately counting the number of such intersections, (iii) computing the intersection of two nonconvex polyhedra, (iv) detecting, counting, or reporting intersections in a set of lines in $\mathbb{R}^3$, and (v) output-sensitive construction of an arrangement of triangles in three dimensions.

Our approach is based on the polynomial partitioning technique.

For example, our ray-shooting algorithm processes a set of n triangles in $\mathbb{R}^3$ into a data structure for answering ray shooting queries amid the given triangles, which uses $O(n^{3/2+\varepsilon})$ storage and preprocessing, and answers a query in $O(n^{1/2+\varepsilon})$ time, for any $\varepsilon > 0$. This is a significant improvement over known results, obtained more than 25 years ago, in which, with this amount of storage, the query time bound is roughly $n^{5/8}$. The algorithms for the other problems have similar performance bounds, with similar improvements over previous results.

We also derive a nontrivial improved tradeoff between storage and query time. Using it, we obtain algorithms that answer m queries on n objects in

$$\max\left\{O(m^{2/3}n^{5/6+\varepsilon} + n^{1+\varepsilon}),\ O(m^{5/6+\varepsilon}n^{2/3} + m^{1+\varepsilon})\right\}$$

time, for any $\varepsilon > 0$, again an improvement over the earlier bounds.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Ray shooting, Three dimensions, Polynomial partitioning, Tradeoff

Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.34

Related Version *Full Version*: https://arxiv.org/abs/2102.07310

Funding *Esther Ezra*: Work partially supported by NSF CAREER under grant CCF:AF-1553354 and by Grant 824/17 from the Israel Science Foundation.
Micha Sharir: Work partially supported by ISF Grant 260/18, by grant 1367/2016 from the German-Israeli Science Foundation (GIF), and by Blavatnik Research Fund in Computer Science at Tel Aviv University.

Acknowledgements We wish to thank Pankaj Agarwal for the useful interaction concerning certain aspects of the range searching problem.

1 Introduction

In this paper we consider several algorithmic problems that involve, explicitly or implicitly, a finite set of lines in three dimensions. The main problems that we consider are:

(i) *Ray shooting amid triangles in three dimensions.* Given a set $\mathcal{T}$ of n triangles in $\mathbb{R}^3$, preprocess $\mathcal{T}$ into a data structure that supports efficient ray-shooting queries, each of which specifies a ray ρ and asks for the first triangle of $\mathcal{T}$ that is hit by ρ, if such a triangle exists.

(ii) *Intersection reporting, emptiness, and approximate counting queries amid triangles in three dimensions.* For a set $\mathcal{T}$ of n triangles in $\mathbb{R}^3$, preprocess $\mathcal{T}$ into a data structure that supports efficient intersection reporting (resp., emptiness) queries, each of which specifies a line, ray, or segment ρ and asks for reporting the triangles of $\mathcal{T}$ that ρ intersects (resp., determining whether such a triangle exists). We want the queries to be output-sensitive, whose cost is a small (sublinear) overhead plus a nearly linear term in the output size k. For approximate counting queries, we want to preprocess $\mathcal{T}$ into a data structure, such that given a query ρ as above, it computes the number of triangles of $\mathcal{T}$ that ρ intersects, up to some prescribed small relative error.

(iii) Compute the intersection of two nonconvex polyhedra. The complexity of the intersection can be quadratic in the complexities of the input polyhedra, and we therefore seek an output-sensitive solution, where the running time is a small (subquadratic) overhead plus a term that is nearly linear in k, where k is the complexity of the intersection.

(iv) Detect, count, or report intersections in a set of lines in 3-space. Again, in the reporting version we seek an output-sensitive solution, as above.

(v) Output-sensitive construction of an arrangement of triangles in three dimensions.

All these problems, or variants thereof, have been considered in several works during the 1990s; see [3, 4, 11, 13, 14, 21, 23] for a sample of these works. See also Pellegrini [24] for a recent comprehensive survey of the state of the art in this area.

Pellegrini [23] presents solutions to some of these problems, including efficient data structures (albeit less efficient than ours) for the ray-shooting problem, and also (a) an output-sensitive algorithm for computing the intersection of two nonconvex polyhedra in time $O(n^{8/5+\varepsilon} + k \log k)$, for any $\varepsilon > 0$, where n is the number of vertices, edges, and facets of the two polyhedra and k is the (similarly defined) complexity of their intersection; (b) an output-sensitive algorithm for constructing an arrangement of n triangles in 3-space in $O(n^{8/5+\varepsilon} + k \log k)$ time, where k is the output size; and (c) an algorithm that, in $O(n^{8/5+\varepsilon})$ expected time, counts all pairs of intersecting lines, in a set of n lines in 3-space.

Background. Algorithmic problems that involve lines in three dimensions have been studied for more than 30 years, covering the problems mentioned above and several others. An early study by McKenna and O'Rourke [22] has developed some of the tools and techniques for tackling these problems. Various techniques for ray shooting, and for the related problems of computing and verifying depth orders and hidden surface removal have been studied in de Berg's dissertation [13], and later by de Berg et al. [14]. Another work that developed some of the infrastructure for these problems is by Chazelle et al. [11], who presented several combinatorial and algorithmic results for problems involving lines in 3-space. Agarwal and Matoušek [3] reduced ray shooting problems, via parametric search, to segment emptiness problems (where the query is a segment and we want to determine whether it intersects any input object), and obtained efficient solutions via this reduction. See also [21] and [4] for studies of some additional and special cases of the ray shooting problem.

Most of the works cited above suffer from the "curse' of the four-dimensionality of (the parametric representation of) lines in space, which leads to algorithms whose complexity is inferior to those obtained in our work. Nevertheless, there are a few instances where better solutions can be obtained, such as in [10, 11] and some other works.

Our results. Using the polynomial partitioning technique of [16, 17], we derive more efficient algorithms for the problems listed above. In our first main result, presented in Section 2, we tackle the ray-shooting problem, and construct a data structure on an input set of n triangles,

which requires $O(n^{3/2+\varepsilon})$ storage and preprocessing, so that a ray shooting query can be answered in $O(n^{1/2+\varepsilon})$ time, for any $\varepsilon > 0$. We then extend the technique, in Section 3, to obtain an equally-efficient data structure for the segment-triangle intersection reporting, emptiness, and approximate counting problems, where in the case of reporting the query time bound has an additional term that is nearly linear in the output size.

These are significant improvements over previous results, which, as already noted, have treated the lines supporting the edges of the input triangles and the line supporting the query ray (or segment) as points or surfaces in a suitable four-dimensional parametric space (in many of the earlier works, lines were actually represented as points on the *Klein quadric* in five-dimensional projective space; see [8, 19, 24, 26]). As a result, the algorithms obtained by these techniques were less efficient.

A weakness, or rather an intriguing peculiarity, of our analysis is that it does not provide a desirably sharp tradeoff between storage and query time. To make this statement more precise, the tradeoff that the earlier solutions provide, say for the ray shooting problem for specificity, is that, for n input triangles and with s storage, for s varying between n and n^4, a ray-shooting query takes $O(n^{1+\varepsilon}/s^{1/4})$ time; see, e.g., [24] (the "4" in the exponent comes from the fact that lines in 3-space are represented as objects in four-dimensional parametric space). Thus, with storage $O(n^{3/2+\varepsilon})$, which is what our solution uses, the query time becomes about $O(n^{5/8})$, considerably weaker than our bound.

An ambitious, and maybe unrealistic goal would be to improve the tradeoff so that the query time is only $O(n^{1+\varepsilon}/s^{1/3})$. (This does indeed coincide with the bound that our main result gives, as the storage that it uses is $O(n^{3/2+\varepsilon})$, but this coincidence only holds for this amount of storage.) Although not achieving this goal, still, combining our technique with the known, aforementioned "4-dimensional" tradeoff, we are able to obtain an "in between" tradeoff, which we present in Section 4, where, with s storage, the query cost is

$$Q(n,s) = \begin{cases} O\left(\frac{n^{5/4+\varepsilon}}{s^{1/2}}\right), & s = O(n^{3/2+\varepsilon}), \\ O\left(\frac{n^{4/5+\varepsilon}}{s^{1/5}}\right), & s = \Omega(n^{3/2+\varepsilon}). \end{cases} \tag{1}$$

Note that this tradeoff contains our bounds $(s, Q) = \big(O(n^{3/2+\varepsilon}), O(n^{1/2+\varepsilon})\big)$, as a special case, that at the extreme ends $s = \Theta(n)$, $s = \Theta(n^4)$, of the range of s we get $Q = O(n^{3/4+\varepsilon})$, $Q = O(n^{\varepsilon})$, respectively,[1] as in the older tradeoff, and that the new tradeoff is better for any in-between value of s. A comparison between the two tradeoffs is illustrated in Figure 1. Our improved tradeoff applies to all the problems studied in this paper: the overall cost of processing m queries with n input objects, including preprocessing cost, is

$$\max\Big\{O(m^{2/3}n^{5/6+\varepsilon} + n^{1+\varepsilon}),\ O(n^{2/3}m^{5/6+\varepsilon} + m^{1+\varepsilon})\Big\}, \tag{2}$$

for any $\varepsilon > 0$; for the output-sensitive problems, this bounds the total overhead cost. The first (resp., second) bound dominates when $n \geq m$ (resp., $n \leq m$).

We then present, in Section 5, extensions of our technique for solving the other problems (iii), (iv) and (v) listed above. In all these applications, our algorithms are output-sensitive for the reporting versions, so that the query time bound, or the full processing cost bound, contains an additional term that is nearly linear in the output size. See Section 5.

Due to lack of space, many details of the above results are omitted in this version. They can all be found in the full version [15].

[1] The actual query time in the older tradeoff, with maximum storage, is $Q = O(\log n)$.

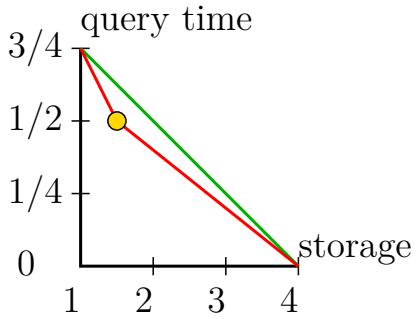


Figure 1 The old tradeoff (green) and the new tradeoff (red). The x-axis is the storage as a function of n, and the y-axis is the query cost. Both axes are drawn in a logarithmic scale.

2 Ray shooting amid triangles

Let $\mathcal{T}$ be a collection of n triangles in $\mathbb{R}^3$. We fix some sufficiently large constant parameter D, and construct a partitioning polynomial f of degree $O(D)$ for $\mathcal{T}$, so that each of the $O(D^3)$ connected components τ of $\mathbb{R}^3 \setminus Z(f)$ (the cells of the partition) is crossed by at most n/D^2 triangle edges. We refer to triangles with an edge that crosses τ as *narrow triangles* (with respect to τ), and refer to the remaining triangles that cross τ (but none of their edges do) as *wide triangles*. We denote the set of narrow (resp., wide) triangles in τ by $\mathcal{N}_\tau$ (resp., $\mathcal{W}_\tau$). The existence of such a partitioning polynomial is implied, as a special case, by the general machinery developed in Guth [16]. An algorithm for constructing f is given in a recent work of Agarwal et al. [2]. It runs in $O(n)$ time, for any constant value of D, where the constant of proportionality depends (polynomially) on D.

For technical reasons, we want to turn any query ray into a bounded segment, and we do it by enclosing all the triangles of $\mathcal{T}$ by a sufficiently large bounding box B_0, and by clipping any query ray to its portion within B_0.

For each (bounded) cell $\tau \subseteq B_0$ of the partition, we take the set $\mathcal{W}_\tau$ of wide triangles in τ, and prepare a data structure for efficient segment-shooting queries into the triangles of $\mathcal{W}_\tau$, by segments that are fully contained in τ. The nontrivial details of this procedure are given in Section 2.1. As we show there, we can construct such a structure with storage and preprocessing $O(|\mathcal{W}_\tau|^{3/2+\varepsilon}) = O(n^{3/2+\varepsilon})$, for any $\varepsilon > 0$ (where the choice of D depends on ε), and each segment-shooting query takes $O(|\mathcal{W}_\tau|^{1/2+\varepsilon}) = O(n^{1/2+\varepsilon})$ time.

The preprocessing then recurses within each such cell τ of the partition, with the set $\mathcal{N}_\tau$ of the narrow triangles in τ. The recursion terminates when the number of input triangles becomes smaller than the constant threshold $n_0 := O(D^2)$, in which case we simply output the list of triangles in the subproblem.

A query, with a ray (now turned into a segment) ρ, emanating from a point q, is answered as follows. We first consider the case where ρ (that is, the line containing ρ) is not fully contained in $Z(f)$, and discuss the (simpler, albeit still involved) case where $\rho \subset Z(f)$, later.

The case where $\rho \not\subset Z(f)$. We use a standard model of algebraic computation, in which computations involving polynomials of constant degree, such as computing (some discrete representation of) their roots, performing comparisons and algebraic computations (of constant degree) with these roots, and so on, can be done exactly in constant time $C(\delta)$, that depends on the degree δ of the polynomial; see, e.g., [6, 7].

Using this model, we first locate the cell of the partition that contains the starting endpoint q of the segment ρ, in constant time (recalling that D is a constant). One way of doing this is to construct the *cylindrical algebraic decomposition (CAD)* of $Z(f)$ (see [12,25]), associate with each cell σ of the CAD the cell of $\mathbb{R}^3 \setminus Z(f)$ that contains it (or an indication that σ is contained in $Z(f)$), and then search with q in the CAD, coordinate by coordinate (see, e.g., [2] for more details). We then find, in constant time, the $t = O(D)$ points of intersection of ρ with $Z(f)$, and sort them into a sequence $P := (p_1, \ldots, p_t)$ in their order along ρ; we assume that $p_t \in \partial B_0$, and ignore the suffix of ρ from p_t onwards. The points in P partition ρ into a sequence of segments, each of which is a connected component of the intersection of ρ with some cell. The first segment is $e_1 = qp_1$, the subsequent segments are $e_2 = p_1p_2$, $e_3 = p_2p_3, \ldots, e_t = p_{t-1}p_t$. We denote by τ_i the cell containing the i-th segment, for $i = 1, \ldots, t$ (a cell can appear several times in this sequence). See Figure 2.

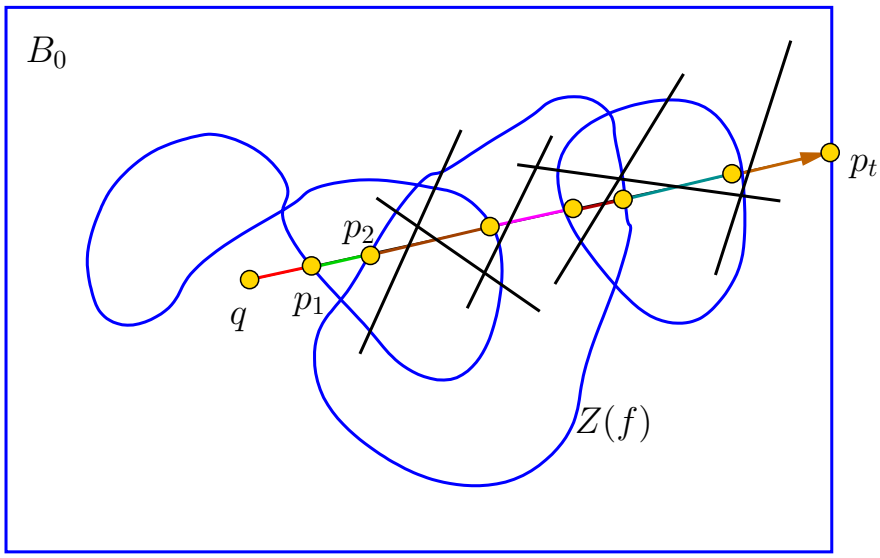


Figure 2 A two-dimensional rendering of the the general structure of the ray-shooting mechanism.

We now process the segments e_i in order. For each segment e_i, let τ_i denote the partition cell that contains e_i. We first perform a ray-shooting (or rather a segment-shooting) query in the structure for $\mathcal{W}_{\tau_i}$ with the segment e_i. As already mentioned (and will be described in Section 2.1), this step can be performed in $O(n^{1/2+\varepsilon})$ time, with $O(n^{3/2+\varepsilon})$ storage and preprocessing, for any $\varepsilon > 0$. We then query with e_i in the substructure recursively constructed for $\mathcal{N}_{\tau_i}$. If at least one of the two queries succeeds, i.e., outputs a point that lies on e_i, we report the point nearest to the starting point of e_i, and terminate the whole query. If both queries fail, we proceed to the next segment e_{i+1} and repeat this step. If the mechanism fails for all the segments, we report that ρ does not hit any triangle of $\mathcal{T}$.

The case where $\rho \subset Z(f)$. We use the cylindrical algebraic decomposition (CAD) of $Z(f)$ (see [12,25]), already discussed earlier. One of its by-products is a *stratification* of $Z(f)$, which is a decomposition of $Z(f)$ into pairwise disjoint relatively open patches of dimensions 0, 1, and 2, called *strata* (each stratum is a cell of the CAD), so that each of the two-dimensional strata is xy-monotone and its relative interior is free of any singularities of $Z(f)$, and $Z(f)$ is the union of the closures of these two-dimensional strata, ignoring possible components of $Z(f)$ of dimension at most 1. We compute the intersection arcs

$\gamma_\Delta := Z(f) \cap \Delta$, for $\Delta \in \mathcal{T}$, and distribute each arc amid the closures of the two-dimensional strata that it traverses. We then project the closure of each two-dimensional stratum σ onto the xy-plane, including the portions of the arcs γ_Δ that the closure contains, and preprocess the resulting collection of $O(n)$ algebraic arcs, each of degree $O(D) = O(1)$, into a planar ray-shooting data structure, whose details are omitted in this version, and can be found in the full version [15].[2] As we show there, we can answer a ray-shooting query in $O(n^{1/2+\varepsilon})$ time, using $O(n^{3/2+\varepsilon})$ storage, for any $\varepsilon > 0$, where the constants of proportionality depend on ε, as does the choice of D. The overall storage complexity, over all the (projected) strata of $Z(f)$, is thus $O(n^{3/2+\varepsilon})$, and the overall query time, over all strata met by the query ray ρ, is $O(n^{1/2+\varepsilon})$, for a larger constant of proportionality (that depends on ε).

The recursion on D terminates when the query ray comes to lie on the zero set of the current partitioning polynomial. We solve the problem in such a recursive instance using a (nonrecursive) procedure, as detailed in the full version [15], and terminate the (current branch of the) recursion. That is, the leaves of the D-recursion tree represent either constant-size subproblems or subproblems on the zero set of the current partitioning polynomial, and the inner nodes represent subproblems of shooting within the partition cells.

Analysis. The correctness of the procedure is fairly easy to establish. Denote by $S(n)$ the maximum storage used by the structure for a set of at most n triangles, and denote by $S_0(n)$ (resp., $S_1(n)$) the maximum storage used by the auxiliary structure for a set of at most n wide triangles in a cell of the partition, as analyzed in Section 2.1 (resp., for a set of at most n intersection arcs on $Z(f)$, which we process for planar ray-shooting (see the full version [15]). Then $S(n)$ obeys the recurrence

$$S(n) = O(D^3)S_0(n) + S_1(n) + O(D^3)S(n/D^2), \tag{3}$$

for $n > n_0$, and $S(n) = O(n)$ for $n \le n_0$, where $n_0 := cD^2$, for a suitable constant $c \ge 1$. We show, in the respective Section 2.1 and the full version [15], that $S_0(n) = O(n^{3/2+\varepsilon})$ and $S_1(n) = O(n^{3/2+\varepsilon})$, for any $\varepsilon > 0$, where both constants of proportionality depend on D and ε, from which one can easily show that the solution of (3) is $S(n) = O(n^{3/2+\varepsilon})$, for a slightly larger, but still arbitrarily small $\varepsilon > 0$; to achieve this bound, we need to take D to be $2^{\Theta(1/\varepsilon)}$, as will follow from our analysis. Regarding the bound on the preprocessing time $T(n)$, we obtain a similar recurrence as in (3), namely,

$$T(n) = O(n) + O(D^3)T_0(n) + T_1(n) + O(D^3)T(n/D^2),$$

where the non-recursive linear term is the time to compute the polynomial f, and $T_0(n)$, $T_1(n)$ are defined in an analogous manner as above, and have similar upper bounds as $S_0(n)$, $S_1(n)$ (see Section 2.1 and the full version [15]).

Similarly, denote by $Q(n)$ the maximum query time for a set of at most n triangles, and denote by $Q_0(n)$ (resp., $Q_1(n)$) the maximum query time in the auxiliary structure for a set of at most n wide triangles in a cell of the partition (resp., for a set of at most n intersection arcs within $Z(f)$, when the query ray lies on $Z(f)$). Then $Q(n)$ obeys the recurrence (recall that the recursion terminates when the query ray lies on the zero set)

$$Q(n) = \max\left\{O(D)Q_0(n) + O(D)Q(n/D^2),\ Q_1(n)\right\}, \tag{4}$$

[2] This specific planar ray-shooting problem, amid constant-degree algebraic arcs, has not received full attention in the past, although several algorithms have been proposed, mostly with suboptimal solutions. Consult, e.g., Table 2 in Agarwal [1]; see also [5, 20].

for $n > n_0$, and $Q(n) = O(n) = O(1)$ for $n \leq n_0$. Again, the analysis in Section 2.1 and the full version [15] shows that $Q_0(n) = Q_1(n) = O(n^{1/2+\varepsilon})$, for any $\varepsilon > 0$ (where the choice of D depends on ε, as above), from which one can easily show, using induction on n, that the solution of (4) is $Q(n) = O(n^{1/2+\varepsilon})$, for a slightly larger but still arbitrarily small $\varepsilon > 0$. The main result of this section is therefore:

▶ **Theorem 1.** *Given a set T of n triangles in three dimensions, and a prescribed parameter $\varepsilon > 0$, we can process T into a data structure of size $O(n^{3/2+\varepsilon})$, in time $O(n^{3/2+\varepsilon})$, so that a ray shooting query amid these triangles can be answered in $O(n^{1/2+\varepsilon})$ time.*

2.1 Ray shooting into wide triangles

Preliminaries. In this subsection we present and analyze our procedure for ray shooting in the set $\mathcal{W}_\tau$ of the wide triangles in a cell τ of the partition. We then present, only in the full version [15], a different approach that yields a procedure for ray shooting within $Z(f)$. Both procedures have the performance bounds stated in Theorem 1. The efficiency of our structures depends on D being a constant, since the constants of proportionality depend polynomially (and rather poorly) on D.

We thus focus now on ray shooting in a set of wide triangles within a three-dimensional cell τ of the partition. To appreciate the difficulty in solving this subproblem, we make the following observation. A simple-minded approach might be to replace each wide triangle $\Delta \in \mathcal{W}_\tau$ by the plane h_Δ supporting it. Denoting the set of these planes as $\mathcal{H}_\tau$, we could then preprocess $\mathcal{H}_\tau$ for ray-shooting queries, each of which specifies a query ray ζ and asks for the first intersection of ζ with the planes of $\mathcal{H}_\tau$. Using standard machinery (see, e.g. [1]), this would result in an algorithm with the performance bounds that we want. However, this approach is problematic, since, even though Δ is wide in τ, h_Δ could intersect τ in several connected components, some of which lie outside Δ. See Figure 3 for an illustration. In such cases, ray shooting amid the planes in $\mathcal{H}_\tau$ is not equivalent to ray shooting amid the triangles of $\mathcal{W}_\tau$, even for rays, or rather portions thereof, that are contained in τ.

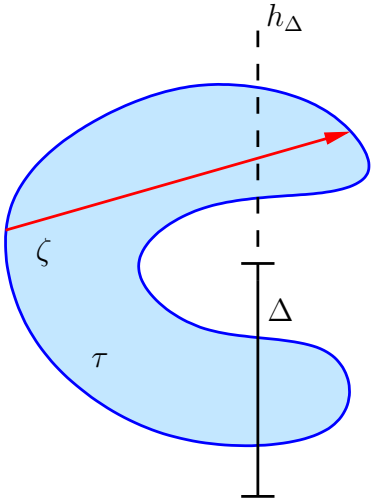


Figure 3 Wide triangles cannot be replaced by their supporting planes for ray shooting within τ.

Our solution is therefore more involved, and proceeds as follows.

Canonical sets of wide triangles. Consider first, for exposition sake, the case where the starting point of the shooting segment lies on $\partial\tau$ (the terminal point always lies on $\partial\tau$). As we will show, for each such segment query, the set of wide triangles in $\mathcal{W}_\tau$ that it intersects can be decomposed into a small collection of precomputed "canonical" subsets, where in each canonical set the wide triangles can be treated as planes (for that particular query segment). The overall size of these sets, over all possible segment queries, is $O(n^{3/2+\varepsilon})$, for any $\varepsilon > 0$.

Actually, to prepare for the complementary case, where the starting point of the query segment lies inside τ, we calibrate our algorithm, so that we control the storage that it uses, and consequently also the query time bound. We introduce a *storage parameter* s, which can range between n and n^2, as a second input to our procedure, and then require that the actual storage and preprocessing cost be both $O(s^{1+\varepsilon})$, for any $\varepsilon > 0$. This relaxed notion of storage offers some simplification in the analysis.

For each $\Delta \in \mathcal{W}_\tau$, let γ_Δ denote the intersection curve of Δ with $\partial\tau$. Note that γ_Δ does not have to be connected – it can have up to $O(D^2)$ connected components, by Harnack's curve theorem [18] (applied on the plane containing Δ). Note also that $\partial\tau$ does not have to be connected, so γ_Δ can have nonempty components on different connected components of $\partial\tau$, as well as several components on the same connected component of $\partial\tau$.

We construct the locus S_τ of points on $\partial\tau$ that are either singular points of $Z(f)$ or points with z-vertical tangency. Since D is constant, S_τ is a curve of constant degree (by Bézout's theorem, its degree is $O(D^2)$). We take a random sample $\mathcal{R}$ of r_0 triangles of $\mathcal{W}_\tau$, where the analysis dictates that we choose $r_0 = D^{\Theta(1/\varepsilon)}$, for the arbitrarily small prescribed $\varepsilon > 0$. Since we have chosen D to be $2^{\Theta(1/\varepsilon)}$, the actual choice of r_0 is $2^{\Theta(1/\varepsilon^2)}$.

Let $\Gamma_\mathcal{R} = \{\gamma_\Delta \mid \Delta \in \mathcal{R}\}$, and let $\mathcal{A}_0 = \mathcal{A}(\Gamma_\mathcal{R} \cup \{S_\tau\})$ denote the arrangement of these curves within $\partial\tau$, together with S_τ. By construction, each face of $\mathcal{A}_0$ is xy-monotone and does not cross any other branch of $Z(f)$ (at a singular point). We partition each face φ of $\mathcal{A}_0$ into pseudo-trapezoids (called trapezoids for short), using a suitably adapted version of a two-dimensional vertical decomposition scheme. Let $\mathcal{A}_0^*$ denote the collection of these trapezoids on $\partial\tau$. The number of trapezoids in $\mathcal{A}_0^*$ is proportional to the complexity of $\mathcal{A}_0$, which is $O_D(r_0^2) = O(1)$.

We assume that the trapezoids are relatively open. For exposition sake, we only handle here two-dimensional trapezoids; lower-dimensional boundary features are handled in the full version [15]. Let ψ_1, ψ_2 be two distinct trapezoids of $\mathcal{A}_0^*$. Let $S(\psi_1, \psi_2)$ denote the collection of all segments e such that one endpoint of e lies in ψ_1, the other endpoint lies in ψ_2, and the relative interior of e is fully contained in the open cell τ. We can parameterize such a segment e by four real parameters, two for the starting endpoint of e and two for its other endpoint. Denote by $\mathcal{F}$ the corresponding (at most) four-dimensional parametric space. Since each of τ, ψ_1, ψ_2 is of constant complexity, $S(\psi_1, \psi_2)$ is a semi-algebraic set in $\mathcal{F}$ of constant complexity, implicitly expressed by the quantified formula

$$S(\psi_1, \psi_2) = \{(p_1, p_2) \mid p_1 \in \psi_1, p_2 \in \psi_2, \text{ and } p_1p_2 \subset \tau\},$$

where p_1p_2 denotes the line-segment connecting p_1 to p_2. Using the singly exponential quantifier-elimination algorithm in [7, Theorem 14.16], we can construct, in $O_D(1)$ time, a quantifier-free semi-algebraic representation of $S(\psi_1, \psi_2)$ of $O_D(1)$ complexity, and we can decompose $S(\psi_1, \psi_2)$ into its connected components, in $O_D(1)$ time as well.

For each segment $e \in S(\psi_1, \psi_2)$, let $\mathcal{T}(e)$ denote the set of all wide triangles of $\mathcal{W}_\tau$ that e crosses. We have the following technical lemma, whose proof can be found in [15].

▶ **Lemma 2.** *In the above notations, each connected component C of $S(\psi_1, \psi_2)$ can be associated with a fixed set $\mathcal{T}_C$ of wide triangles of $\mathcal{W}_\tau$, none of which crosses $\psi_1 \cup \psi_2$, so that, for each segment $e \in C$, $\mathcal{T}_C \subseteq \mathcal{T}(e)$, and each triangle in $\mathcal{T}(e) \setminus \mathcal{T}_C$ crosses either ψ_1 or ψ_2.*

The collection of all these sets $\mathcal{T}_C$, over all connected components C, and all pairs of trapezoids (ψ_1, ψ_2), is part of the whole output collection of canonical sets over τ; the rest of this collection is constructed recursively over the trapezoids ψ of $\mathcal{A}_0^*$.

The algorithm. For each trapezoid ψ of $\mathcal{A}_0^*$, the *conflict list* K_ψ of ψ is the set of all wide triangles that cross ψ. By standard random sampling arguments [9], with high probability, the size of each K_ψ is $O_D\left(\frac{n}{r_0}\log r_0\right)$. (Special cases, involving lower-dimensional boundary features and triangles fully containing trapezoids, are handled in the full version [15].)

For every pair of trapezoids ψ_1, ψ_2, we compute $S(\psi_1, \psi_2)$ and decompose it into its connected components. We pick some arbitrary but fixed segment e_0 from each component C, compute the set $\mathcal{T}(e_0)$ of the wide triangles that cross e_0, and remove from it any triangle that crosses $\psi_1 \cup \psi_2$, thereby obtaining the set $\mathcal{T}_C$. All this takes $O_D(r_0^4 n) = O_D(n)$ time, and the overall size of the produced canonical sets is also $O_D(n)$.

Let s be the storage parameter associated with the problem, as defined earlier, and recall that we require that $n \le s \le n^2$. Each canonical set $\mathcal{T}_C$ is preprocessed into a data structure that supports ray shooting in the set of *planes* $\mathcal{H}_C = \{h_\Delta \mid \Delta \in \mathcal{T}_C\}$, where h_Δ is the plane supporting Δ. We construct these structures so that they use $O(s^{1+\varepsilon})$ storage (and preprocessing), for any $\varepsilon > 0$, and a query takes $O(n \operatorname{polylog}(n)/s^{1/3})$ time (see, e.g., [1]).

We now recurse on each conflict list K_ψ, over all trapezoids ψ of $\mathcal{A}_0^*$. Each subproblem uses the same parameter r_0, but now the storage parameter that we allocate to each subproblem is only s/r_0^2. We keep recursing until we reach conflict lists of size close to n^2/s. More precisely, after j levels of recursion, we get a total of at most $(c_0 r_0^2)^j = c_0^j r_0^{2j}$ subproblems, each involving at most $\left(\frac{c_1 \log r_0}{r_0}\right)^j n$ wide triangles, for some constants c_0, c_1 that depend on D, and thus on ε, but are considerably smaller than $r_0 = D^{\Theta(1/\varepsilon)}$.

We stop the recursion at the first level j^* at which $(c_1 r_0 \log r_0)^{j^*} \ge s/n$. As a result, we have ${r_0}^{j^*} \le s/n$, and we get $c_0^{j^*} {r_0}^{2j^*} = O(s^2/n^{2-\varepsilon})$ subproblems, for any $\varepsilon > 0$, where the choice of D (and therefore also of c_0, c_1 and r_0) depends, as above, on ε. With a suitable choice of D, each of these subproblems involves at most

$$\left(\frac{c_1 \log r_0}{r_0}\right)^{j^*} n = \left(\frac{(c_1 \log r_0)^2}{c_1 r_0 \log r_0}\right)^{j^*} n \le (c_1 \log r_0)^{2j^*} \cdot \frac{n^2}{s} = \frac{n^{2+\varepsilon}}{s}$$

triangles, for any $\varepsilon > 0$. Hence the overall size of the inputs and of the canonical sets, at all recursive subproblems, is $O\left(\frac{s^2}{n^{2-\varepsilon}}\right) \cdot \frac{n^{2+\varepsilon}}{s} = O(sn^{2\varepsilon}) = O(s^{1+\varepsilon})$, for a slightly larger $\varepsilon > 0$.

Note that the canonical sets that we encounter when querying with a fixed segment e are not necessarily pairwise disjoint. This is because the sets K_{ψ_1} and K_{ψ_2} are not necessarily disjoint (they are disjoint of $\mathcal{T}_C$, though). This does not pose a problem for ray shooting queries, but will be problematic for *counting queries*; see Section 3.

At the bottom of the recursion, each subproblem contains at most $n^{2+\varepsilon}/s$ wide triangles, which we merely store in the structure. As just calculated, the overall storage that this requires is $O(s^{1+\varepsilon})$, for a slightly larger ε, as above. We obtain the following recurrence for the overall storage $S(N_W, s_W)$ for the structure constructed on N_W wide triangles, where s_W denotes the storage parameter allocated to the structure (at the root $N_W = n$, $s_W = s$).

$$S(N_W, s_W) = \left\{ \begin{array}{ll} O_D(r_0^4 s_W^{1+\varepsilon}) + c_0 r_0^2 S\left(\frac{c_1 N_W \log r_0}{r_0}, \frac{s_W}{r_0^2}\right) & \text{for } N_W \ge n^{2+\varepsilon}/s, \\ O(N_W) & \text{for } N_W < n^{2+\varepsilon}/s. \end{array} \right\}$$

Throughout the recursion we have $N_W \le s_W \le N_W^2$ (see the full version for details).

Unfolding the first recurrence up to the terminal level j^*, where $N_W < n^{2+\varepsilon}/s$, the sum of the nonrecursive overhead terms $O_D(r_0^4 s_W^{1+\varepsilon})$, over all nodes at a fixed level j, is

$$c_0^j r_0^{2j} \cdot O\left(\frac{s_W^{1+\varepsilon}}{r_0^{2j(1+\varepsilon)}}\right) = O\left(\frac{c_0^j}{r_0^{2j\varepsilon}} s_W^{1+\varepsilon}\right) = O\left(s_W^{1+\varepsilon}\right),$$

by the choice of r_0. Hence, starting the recurrence at $(N_W, s_w) = (n, s)$, the overall contribution of the overhead terms (over the logarithmically many levels) is $O(s^{1+\varepsilon})$, for a slightly larger ε. At the bottom of recurrence, we have, as already noted, $O(s^2/n^{2-\varepsilon})$ subproblems, each with at most $O(n^{2+\varepsilon}/s)$ triangles, so the sum of the terms N_W at the bottom of recurrence is also $O(s^{1+\varepsilon})$. In other words, the overall storage used by the data structure is $O(s^{1+\varepsilon})$. Using similar considerations, one can show that the overall preprocessing time is $O(s^{1+\varepsilon})$ as well, since the time obeys essentially the same recurrence.

Answering a query. To perform a query with a segment e that starts at a point a (that lies anywhere inside τ), we extend e from a backwards, find the first intersection point a' of the resulting backward ray with $\partial\tau$, and denote by e' the segment that starts at a' and contains e. See Figure 4 for an illustration. This takes $O_D(1)$ time. This step is vacuous when e starts on $\partial\tau$, in which case we have $e' = e$.

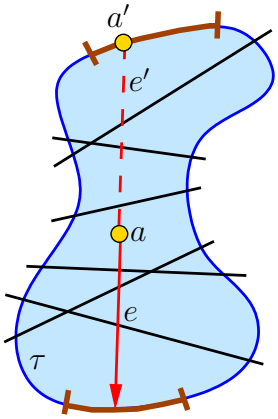


Figure 4 Segment shooting from inside the cell τ: Extending the segment backwards and the resulting canonical set of triangles.

We find the pair of trapezoids ψ_1, ψ_2 that contain the endpoints of e', find the connected component $C \subseteq S(\psi_1, \psi_2)$ that contains e', and retrieve the canonical set $\mathcal{T}_C$. We then perform segment shooting along e from a in the structure constructed for $\mathcal{H}_C$, and then continue recursively in the subproblems for K_{ψ_1} and K_{ψ_2}. We output the triangle that e hits nearest to a, or else report that e does not hit any wide triangle inside τ. See the full version [15] for the case where both endpoints of e' lie in the same trapezoid ψ.

The correctness of the procedure follows from the fact that e' intersects all the triangles of $\mathcal{T}_C$, and thus replacing these triangles by their supporting planes cannot produce any new (false) intersection of any of these triangles with e, and any other wide triangle that e hits must belong to $K_{\psi_1} \cup K_{\psi_2}$.

The query time $Q(N_W, s_W)$ satisfies the recurrence

$$Q(N_W, s_W) = \left\{ \begin{array}{ll} O_D(1) + O\left(\frac{N_W \operatorname{polylog}(N_W)}{s_W^{1/3}}\right) + 2Q\left(\frac{c_1 N_W \log r_0}{r_0}, \frac{s_W}{r_0^2}\right) & \text{for } N_W \geq n^{2+\varepsilon}/s, \\ O(N_W) & \text{for } N_W < n^{2+\varepsilon}/s. \end{array} \right\}.$$

Unfolding the first recurrence, we see that when we pass from some recursive level to the next one, we get two descendant subproblems from each recursive instance, and the term $N_W \text{polylog}(N_W)/s_W^{1/3}$ is replaced in each of them by the (upper bound) term

$$\frac{\frac{c_1 N_W \log r_0}{r_0}}{\left(\frac{s_W}{r_0^2}\right)^{1/3}} \cdot \text{polylog}(N_W) = \frac{c_1 \log r_0}{r_0^{1/3}} \cdot \frac{N_W \text{polylog}(N_W)}{s_W^{1/3}}.$$

Hence the overall bound for the nonrecursive overhead terms in the unfolding, starting from $(N_W, s_W) = (n, s)$, is at most

$$O\left(\sum_{j \geq 0} \left(\frac{2c_1 \log r_0}{r_0^{1/3}}\right)^j\right) \cdot \frac{n \text{ polylog}(n)}{s^{1/3}} = O\left(\frac{n \text{ polylog}(n)}{s^{1/3}}\right),$$

provided that r_0 is sufficiently large. Adding the cost at the 2^{j^*} subproblems at the bottom level j^* of the recursion, where the cost of each subproblem is at most $n^{2+\varepsilon}/s$, gives an overall bound for the query time of

$$Q(n, s) = O\left(\frac{n \text{ polylog}(n)}{s^{1/3}} + \frac{n^{2+\varepsilon}}{s}\right). \tag{5}$$

Starting with $s = n^{3/2}$, the query time is $O(n^{1/2+\varepsilon})$. We thus obtain

▶ **Proposition 3.** *For a (bounded) cell τ of the polynomial partition, and a set $\mathcal{W}$ of n wide triangles in τ, one can construct a data structure of size and preprocessing cost $O(n^{3/2+\varepsilon})$, so that a segment-shooting query within τ, from any starting point, can be answered in $O(n^{1/2+\varepsilon})$ time, for any $\varepsilon > 0$.*

See the full version [15] for the case where the query ray lies in $Z(f)$. We show:

▶ **Proposition 4.** *For a partitioning polynomial f of sufficiently large constant degree, and a set $\mathcal{W}$ of n triangles, one can construct a data structure of size and preprocessing cost $O(n^{3/2+\varepsilon})$, so that a segment-shooting query with a segment that lies in $Z(f)$, can be answered in $O(n^{1/2+\varepsilon})$ time, for any $\varepsilon > 0$.*

As already concluded, Propositions 3 and 4 complete the proof of Theorem 1.

3 Segment-triangle intersection reporting, emptiness, and approximate counting queries

We extend the technique presented in Section 2 to answer intersection reporting queries amid triangles in $\mathbb{R}^3$. Here too we have a set $\mathcal{T}$ of n triangles in $\mathbb{R}^3$, and our goal is to preprocess $\mathcal{T}$ into a data structure that supports efficient intersection queries, each of which specifies a line, ray or segment ρ and asks for reporting the triangles of $\mathcal{T}$ that ρ intersects. In particular, this data structure also supports *segment emptiness* queries, in which we want to determine whether the query segment meets any input triangle. We obtain the following result, whose proof is given in the full version [15].

▶ **Theorem 5.** *Given a collection of n triangles in three dimensions, and a prescribed parameter $\varepsilon > 0$, we can process the triangles into a data structure of size $O(n^{3/2+\varepsilon})$, in time $O(n^{3/2+\varepsilon})$, so that a segment-intersection reporting (resp., emptiness) query amid these triangles can be answered in $O(n^{1/2+\varepsilon} + k \log n)$ (resp., $O(n^{1/2+\varepsilon})$) time, where k is the number of triangles that the query segment crosses.*

Unfortunately, for technical reasons, the method does not extend to segment-triangle intersection (exact) *counting* queries, in which we want to find the (exact) number of triangles that intersect a query segment (or a line or a ray). Briefly, this arises because the canonical sets that a query segment collects are not necessarily pairwise disjoint. As a "compensation", we present a partial solution, which supports queries that approximately count the number of intersections, up to any prescribed relative error $\varepsilon > 0$. Specifically, we obtain the following result, whose proof is given in the full version [15].

▶ **Theorem 6.** *Given a collection of n triangles in three dimensions, and prescribed parameters $\varepsilon, \delta > 0$, where $\delta = \omega(1/n^{\varepsilon})$, we can process the triangles, using random sampling, into a data structure of size $O(n^{3/2+\varepsilon})$, in time $O(n^{3/2+\varepsilon})$, so that, for a query segment e, the number of intersections of e with the input triangles can be approximately computed, up to a relative error of $1 \pm \delta$, with very high probability, in $O(n^{1/2+\varepsilon})$ time.*

4 Tradeoff between storage and query time

In this section, spelled out in the full version [15], we extend the technique in Sections 2 and 3 to obtain a tradeoff between storage (and preprocessing) and query time. A similar tradeoff holds for the problems in Section 5.

Briefly, consider the ray-shooting structure of Section 2, and let s be the storage parameter that we allocate to it, which now satisfies $n \leq s \leq n^4$. We modify the procedure for ray shooting inside a cell τ by (i) stopping the r_0-recursion at some earlier "premature" level, and (ii) modifying the structure at the bottom of recursion so that it uses the (weaker) ray-shooting technique of Pellegrini [23], instead of a brute-force scanning of the triangles (the current cost of $O(n^2/s)$ is too expensive when s is small). With additional care, we obtain the performance bounds (1) and (2) announced in the introduction.

5 Other applications

5.1 Detecting, counting or reporting line intersections in $\mathbb{R}^3$

It is more convenient, albeit not necessary, to consider the bichromatic version of the problem, in which we are given a set R of n red lines and a set B of n blue lines in $\mathbb{R}^3$, and the detection problem asks whether there exists a pair of intersecting lines in $R \times B$.

An algorithm that solves this problem in $O(n^{3/2+\varepsilon})$ time is easily obtained by regarding the problem as a special degenerate (and much simpler) instance of the ray shooting problem, in which we regard the, say red lines as degenerate triangles (unbounded and of zero area), construct the data structure of Section 2 and query it with each of the blue lines. There exists a red-blue pair of intersecting lines if and only if at least one query has a positive outcome – the corresponding blue query line hits a red line.

Since there are no wide triangles in this special variant, there is no need to construct the auxiliary data structure for wide triangles, as in Section 2.1, and we simply construct the recursive hierarchy of polynomial partitions, with the subset of red lines associated with each cell in each subproblem. A blue query line ℓ is propagated through the cells that it crosses until it reaches bottom-level cells, and we check, in each such cell, whether ℓ intersects any of the red lines associated with the cell.

Handling lines that lie fully in the zero set $Z(f)$ is also an easy task (which can be performed using planar segment-intersection range searching, which also supports counting queries); further details are omitted.

Both correctness and runtime analysis follow easily, as special and simpler instances of the analysis in Section 2. Note that here we do not face the issue of non-disjointness of canonical sets of wide triangles, which has prevented us from extending the technique to segment-triangle intersection counting problems; see Section 3.

5.2 Computing the intersection of two polyhedra

Let K_1 and K_2 be two polyhedra in 3-space, not necessarily convex, each with n edges (so the number of vertices and faces of each of them is $O(n)$). The goal is to compute their intersection $K := K_1 \cap K_2$ in an output-sensitive manner. We note that computing the union $K_1 \cup K_2$ can be done using a very similar approach, within the same time bound.

Details of the procedure are given in the full version [15]. Briefly, the main step is to compute all the vertices of K, each of which is either a vertex of K_1 or of K_2, or an intersection of an edge of one polyhedron and a face of the other. The vertices of the latter type are readily obtain by applying the segment intersection reporting algorithm of Section 3, twice, with the (triangulated) faces of one polyhedron as input and the edges of the other as queries. See [15] for the complete algorithm, which yields:

▶ **Corollary 7.** *Given two arbitrary polyhedra K_1 and K_2 each of complexity $O(n)$, we can compute $K_1 \cap K_2$ in time $O(n^{3/2+\varepsilon} + k \log k)$, where k is the size of the intersection.*

As discussed in the introduction, the overhead term of Pellegrini [23] was $O(n^{8/5+\varepsilon})$.

5.3 Output-sensitive construction of an arrangement of triangles

Let $\mathcal{T}$ be a set of n possibly intersecting triangles in $\mathbb{R}^3$, let $\mathcal{A} = \mathcal{A}(\mathcal{T})$ denote their arrangement, and let k denote its complexity, which, as in Section 5.2, we measure by the number of its vertices, as the number of its other features (edges, faces, and cells) is proportional to k. The goal is to construct $\mathcal{A}$ in an output-sensitive manner with a small, subquadratic overhead. Pellegrini [23] gave such an algorithm that runs in $O(n^{8/5+\varepsilon} + k \log n)$, and the algorithm that we present here reduces the overhead to $O(n^{3/2+\varepsilon})$ time.

As in the previous subsection, we focus on the main step of the algorithm that constructs the features of $\mathcal{A}$ (vertices, edges, and faces) on each triangle of $\mathcal{T}$. The other, simpler complementary steps are discussed in the full version.

Fix a triangle $\Delta \in \mathcal{T}$. We first construct the set of intersection segments $\Delta \cap \Delta'$, for $\Delta' \in \mathcal{T} \setminus \{\Delta\}$. We observe that, for any such segment $e = \Delta \cap \Delta'$, each endpoint of e is either a vertex of Δ, or an intersection of an edge of one triangle with the other triangle.

We therefore take the collection of the $3n$ edges of the triangles of $\mathcal{T}$, and, for each such edge e, apply Theorem 5, which reports all k_e triangles that e meets. This identifies all the intersection segments $\Delta \cap \Delta'$. We then take all the intersection segments within a fixed triangle Δ, and run a sweepline procedure within Δ to obtain the portion of $\mathcal{A}$ on Δ. Gluing these portions to each other, and some additional steps, complete the construction of $\mathcal{A}$.

6 Conclusion

In this paper we have managed to improve the performance of ray shooting amid triangles in three dimensions, as well as of several related problems. The improvement is based on the polynomial partitioning technique of Guth. The improvement is most significant when the storage is about $n^{3/2}$ and the query takes about $n^{1/2}$ time, but one gets an improvement for all values of the storage between n and n^4, except at the very ends of this range. This is a significant improvement, the first in nearly 30 years, in this basic problem.

There are several open questions that our work raises. First, can we improve our tradeoff for all values of storage, beyond the special values of $O(n^{3/2+\varepsilon})$ storage and $O(n^{1/2+\varepsilon})$ query time? Ideally, can we obtain query time of $O(n^{1+\varepsilon}/s^{1/3})$, with s storage, as in the case of ray shooting amid planes? Alternatively, can one establish a lower-bound argument that shows the limitations of our technique?

Another open issue follows from our current inability to extend the technique to counting queries, due to the fact that the canonical sets that we collect during a query are not necessarily pairwise disjoint. It would be interesting to obtain such an extension, or, alternatively, to establish a gap between the performances of the counting and reporting versions of the segment intersection query problem.

Finally, could one obtain similar bounds for non-flat input objects? for shooting along non-straight curves? It would also be interesting to find additional applications of the general technique developed in this paper.

References

1 P. K. Agarwal. Simplex range searching and its variants: A review. In J. Nešetřil M. Loebl and R. Thomas, editors, *Journey through Discrete Mathematics: A Tribute to Jiří Matoušek*, pages 1–30. Springer Verlag, Berlin-Heidelberg, 2017.

2 P. K. Agarwal, B. Aronov, E. Ezra, and J. Zahl. An efficient algorithm for generalized polynomial partitioning and its applications. In *Proc. Internat. Sympos. on Computational Geometry*, pages 5:1–5:14, 2019. Also in `arXiv:1812.10269`.

3 P. K. Agarwal and J. Matoušek. Ray shooting and parameric search. *SIAM J. Comput.*, 22:794–806, 1993.

4 P. K. Agarwal and M. Sharir. Ray shooting amidst convex polyhedra and polyhedral terrains in three dimensions. *SIAM J. Comput.*, 25:100–116, 1996.

5 P. K. Agarwal, M. van Kreveld, and M. Overmars. Intersection queries in curved objects. *J. Algorithms*, 15:229–266, 1993.

6 B. Aronov, E. Ezra, and J. Zahl. Constructive polynomial partitioning for algebraic curves in R^3 with applications. *SIAM J. Comput.*, 49:1109–1127, 2020. Also in *Proc. Sympos. on Discrete Algorithms (SODA)*, 2019, 2636–2648. Also in `arXiv:1904.09526`.

7 S. Basu, R. Pollack, and M.-F. Roy. *Algorithms in Real Algebraic Geometry*. Algorithms and Computation in Mathematics 10. Springer-Verlag, Berlin, 2003.

8 O. Bottema and B. Roth. *Theoretical Kinematics*. Dover, New York, 1990.

9 B. Chazelle, H. Edelsbrunner, L. Guibas, and M. Sharir. A singly exponential stratification scheme for real semi-algebraic varieties and its applications. *Theoretical Computer Science*, 84:77–105, 1991.

10 B. Chazelle, H. Edelsbrunner, L. Guibas, and M. Sharir. Algorithms for bichromatic line segment problems and polyhedral terrains. *Algorithmica*, 11:116–132, 1994.

11 B. Chazelle, H. Edelsbrunner, L. J. Guibas, M. Sharir, and J. Stolfi. Lines in space: Combinatorics and algorithms. *Algorithmica*, 15:428–447, 1996.

12 G. E. Collins. Quantifier elimination for the elementary theory of real closed fields by cylindrical algebraic decomposition. In *Proc. 2nd GI Conf. Automata Theory and Formal Languages*. Springer LNCS 33, 1975.

13 M. de Berg. *Ray Shooting, Depth Orders and Hidden Surface Removal*. Lecture Notes Comput. Sci., 703. Springer Verlag, Berlin, 1993.

14 M. de Berg, D. Halperin, M. Overmars, J. Snoeyink, and M. van Kreveld. Efficient ray shooting and hidden surface removal. *Algorithmica*, 12:30–53, 1994.

15 E. Ezra and M. Sharir. On ray shooting for triangles in 3-space and related problems. `arXiv:2102.07310`.

16 L. Guth. Polynomial partitioning for a set of varieties. *Math. Proc. Camb. Phil. Soc.*, 159:459–469, 2015. Also in `arXiv:1410.8871`.

17 L. Guth and N.H̃. Katz. On the erdHos distinct distances problem in the plane. *Annals Math.*, 181:155–190, 2015. Also in `arXiv:1011.4105`.

18 C. G. A. Harnack. Über die vielfaltigkeit der ebenen algebraischen kurven. *Math. Ann.*, 10:189–199, 1876.

19 K. Hunt. *Kinematic Geometry of Mechanisms*. Oxford, 1990.

20 V. Koltun. Segment intersection searching problems in general settings. *Discrete Comput. Geom.*, 30:25–44, 2003.

21 J. Matoušek and O. Schwarzkopf. On ray shooting in convex polytopes. *Discrete Comput. Geom.*, 10:215–232, 1993.

22 M. McKenna and J. O'Rourke. Arrangements of lines in 3-space: A data structure with applications. In *Proc. 4th ACM Sympos. Computational Geometry*, pages 371–380, 1988.

23 M. Pellegrini. Ray shooting on triangles in 3-space. *Algorithmica*, 9:471–494, 1993.

24 M. Pellegrini. Ray shooting and lines in space. In J. O'Rourke J. E. Goodman and C. D. Tóth, editors, *Handbook on Discrete and Computational Geometry*, chapter 41, pages 1093–1112. CRC Press, Boca Raton, Florida, Third Edition, 2017.

25 J.T. Schwartz and M. Sharir. On the piano movers' problem: Ii. general techniques for computing topological properties of real algebraic manifolds. *Advances in Appl. Math.*, 4:298–351, 1983.

26 J. Stolfi. *Oriented Projective Geometry*. Academic Press, New York, 1991.

On Rich Lenses in Planar Arrangements of Circles and Related Problems

Esther Ezra
School of Computer Science, Bar Ilan University, Ramat Gan, Israel

Orit E. Raz
Institute of Mathematics, Hebrew University, Jerusalem, Israel

Micha Sharir
School of Computer Science, Tel Aviv University, Israel

Joshua Zahl
Department of Mathematics, University of British Columbia, Vancouver, Canada

Abstract

We show that the maximum number of pairwise non-overlapping k-rich lenses (lenses formed by at least k circles) in an arrangement of n circles in the plane is $O(n^{3/2}\log(n/k^3)k^{-5/2}+n/k)$, and the sum of the degrees of the lenses of such a family (where the degree of a lens is the number of circles that form it) is $O(n^{3/2}\log(n/k^3)k^{-3/2}+n)$. Two independent proofs of these bounds are given, each interesting in its own right (so we believe). We then show that these bounds lead to the known bound of Agarwal et al. (JACM 2004) and Marcus and Tardos (JCTA 2006) on the number of point-circle incidences in the plane. Extensions to families of more general algebraic curves and some other related problems are also considered.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Lenses, Circles, Polynomial partitioning, Incidences

Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.35

Related Version *Full Version*: https://arxiv.org/abs/2012.04204

Funding *Esther Ezra*: Work partially supported by NSF CAREER under grant CCF:AF-1553354 and by Grant 824/17 from the Israel Science Foundation.
Micha Sharir: Work partially supported by ISF Grant 260/18, by grant 1367/2016 from the German-Israeli Science Foundation (GIF), and by Blavatnik Research Fund in Computer Science at Tel Aviv University.
Joshua Zahl: Work supported by an NSERC Discovery Grant.

1 Introduction

Let C be a set of circles in the plane. A *lens* in the arrangement $\mathcal{A}(C)$ consists of a pair of distinct points p, q and a set of circles $C' \subset C$, each of which contain p and q. We will denote a lens by $\lambda_{p,q}(C')$. We say that two lenses $\lambda_{p,q}(C')$ and $\lambda_{s,t}(C'')$ are *overlapping* if there is a circle $c \in C' \cap C''$ so that the shorter arc of c containing p and q intersects the shorter arc of c containing s and t.[1] If two lenses are not overlapping, we call them *non-overlapping*. Finally, the *degree* of a lens $\lambda_{p,q}(C')$ is the cardinality of C', and we say a lens is k-*rich* if it has degree at least k.

[1] This definition requires a small modification if either p, q or s, t are antipodal points of c; if p, q are antipodal points of c, we say the corresponding lens overlaps every lens that contains c.

In this paper, we will be concerned with bounding the maximum size of a collection of pairwise non-overlapping k-rich lenses determined by a set of n circles in the plane. As we will see below, this question is closely related to the problem of *lens cutting*, which has a host of applications in combinatorial geometry; chief among these is the problem of obtaining incidence bounds for points and circles in the plane.

In [8] (sharpening a bound earlier obtained in [1, 3]) Marcus and Tardos proved that if C is a set of n circles, then any set of pairwise non-overlapping 2-rich lenses in C has cardinality $O(n^{3/2}\log n)$. Using standard random sampling techniques, this implies that any set of pairwise non-overlapping k-rich lenses has cardinality $O\big(\frac{n^{3/2}\log(n/k)}{k^{3/2}}\big)$. Our main result considerably improves this bound.

▶ **Theorem 1.** *Let C be a set of n circles in the plane, let $k \geq 2$, and let Λ be a set of pairwise non-overlapping k-rich lenses. Then $|\Lambda| = O\left(\frac{n^{3/2}\log(n/k^3)}{k^{5/2}} + \frac{n}{k}\right)$, and the sum of the degrees of the lenses in Λ is $O\left(\frac{n^{3/2}\log(n/k^3)}{k^{3/2}} + n\right)$.*

As mentioned, Theorem 1 was proved for the case $k = O(1)$ by Marcus and Tardos [8]. When k is large, we will show that Theorem 1 can be recast as an incidence problem between points and lines in $\mathbb{R}^3$. Crucially, we will show that only a few incidences of the type we analyze can occur inside any plane, and this will allow us to use a variant of Guth and Katz's point-line incidence bound from [7] to prove Theorem 1 when $k \geq n^{1/3}$. The details of this argument will be discussed in Section 2.

For intermediate values of k, we will give two proofs of Theorem 1. The first proof yields the bounds stated above, and the second proof gives a sightly weaker bound, in which the $\log(n/k^3)$ factor is weakened to $\operatorname{polylog} n$. Both of these proofs will use polynomial partitioning to divide the arrangement C of circles into smaller sub-arrangements. This breaks the problem of estimating $|\Lambda|$ into many smaller sub-problems, with smaller corresponding parameters n' (for the number of circles) and k' (for the richness). In the first proof, we will construct our partitioning so that in each sub-problem we have $k' = 2$, while in the second proof we will construct our partitioning so that in each sub-problem we have[2] $k' = (n')^{1/3}$.

▶ Remark 2. When $k \geq n^{1/3}\log^{2/3} n$, Theorem 1 states that $|\Lambda| = O(n/k)$. This bound is tight, since we can choose C to be a union of n/k sets of circles, where each set of circles has cardinality k and the circles in each set contain a common pair of points. For smaller values of k we conjecture that the bound in Theorem 1 is not tight.

▶ Remark 3. Theorem 1 implies that the circles in C can be cut into $O\left(\frac{n^{3/2}\log(n/k^3)}{k^{3/2}} + n\right)$ arcs, so that no pair of points is contained in k of the arcs, i.e., the resulting collection of arcs do not form any k-rich lens. Combining this observation with a variant of Székely's crossing-lemma technique [12], yields a new proof that the number of incidences between m points and n circles in the plane is (see Section 5):

$$O\left(m^{2/3}n^{2/3} + m^{6/11}n^{9/11}\log^{2/11} n + m + n\right).$$

This bound was first proved in [1, 8]. The point-circle incidence problem is among the most basic problems in incidence geometry, and has been studied intensively during the first half of the 2000's [1, 3, 8], culminating in the bound above. This bound is strongly suspected not to be tight for $n^{1/3} \leq m < n^{5/4}\log^{3/2} n$ (which is the range where the second term dominates),

[2] To simplify the presentation we ignore, throughout the paper, the issue of rounding non-integer values, and regard any such value as being rounded to the nearest integer.

but no improvement has been found in the last 15 years. While our result also does not yield an improvement, it provides a new proof (two proofs as a matter of fact), and we hope that this development will spur efforts to improve the above bound.

2 Preliminaries: The Case of Large or Small k

In this section we will prove Theorem 1 when k is small or $k \geq n^{1/3}$. As discussed above, when k is small (smaller than some constant) then the result immediately follows from [8].

▶ **Theorem 4** (Marcus and Tardos [8]). *Let C be a set of n circles in the plane, let $k \geq 2$, and let Λ be a set of pairwise non-overlapping k-rich lenses in C. Then Λ has cardinality $O(n^{3/2}\log n)$, and the sum of the degrees of the lenses in Λ is also $O(n^{3/2}\log n)$.*

When $k \geq n^{1/3}$, Theorem 1 will follow from a variant of Guth and Katz's point-line incidence bound [7]. Before stating this result we will need to introduce some additional notation. In what follows, C will be a set of circles in the plane, $k \geq 2$, and Λ will be a set of pairwise non-overlapping k-rich lenses in C. We define $\deg(\Lambda)$ to be the sum of the degrees of the lenses in Λ. We say that a circle $c \in C$ *participates* in a lens $\lambda_{p,q}(C') \in \Lambda$ if $c \in C'$.

We identify each circle c, with center (x, y) and radius r, with the point

$$c^* = (x, y, r^2 - x^2 - y^2)$$

in $\mathbb{R}^3$. We define $C^* = \{c^* \mid c \in C\}$, and identify each point $p = (p_x, p_y) \in \mathbb{R}^2$ with the plane

$$p^* = \{(x, y, z) \mid z = -2p_x x - 2p_y y + (p_x^2 + p_y^2)\}.$$

Observe that $(x, y, r^2 - x^2 - y^2) \in p^*$ if and only if $(x - p_x)^2 + (y - p_y)^2 = r^2$, i.e. the point p is contained in the circle centered at (x, y) of radius r. We identify each lens $\lambda = \lambda_{p,q}(C')$ with the line $\lambda^* = p^* \cap q^*$. Note that the lines $p^* \cap q^*$ and $s^* \cap t^*$ coincide if and only if $\{p, q\} = \{s, t\}$, and therefore our setting does not contain coinciding lines.[3] Define

$$\Lambda^* = \{\lambda^* \mid \lambda \in \Lambda\}.$$

For technical reasons, it will be convenient to require that no two distinct lenses in Λ share the same pair $\{p, q\}$ of endpoints, and thus the map $\lambda_{p,q}(C') \mapsto \lambda^*_{p,q}$ is injective on Λ, i.e., $|\Lambda^*| = |\Lambda|$. As we will see, this additional assumption is harmless, and we will discuss it briefly in Remark 10 below. We define ("nov" is an abbreviation for "non-overlapping")

$$I_{\text{nov}}(\Lambda) = \{(c^*, \lambda^*) \mid c \in C,\ \lambda \in \Lambda,\ c \text{ participates in } \lambda\}.$$

If the sets C and Λ are apparent from the context, we write I_{nov} in place of $I_{\text{nov}}(\Lambda)$. Note that $|I_{\text{nov}}| = \deg(\Lambda)$. If $(c^*, \lambda^*) \in I_{\text{nov}}$ then $c^* \in \lambda^*$. Thus $I_{\text{nov}} \subset I(C^*, \Lambda^*)$, where $I(C^*, \Lambda^*)$ denotes the set of all incidences between the points of C^* and the lines of Λ^*. Note that I_{nov} may be a proper subset of $I(C^*, \Lambda^*)$ due to the pairwise non-overlapping property of the lenses in Λ. That is, a circle c might pass through the vertices p, q of λ but it does not participate in λ because there is another lens $\lambda' \in \Lambda$, in which c does participate, so that the arc of c in λ' overlaps its arc between p and q; see Figure 1 for an illustration.

[3] This is because all planes of the form p^* are tangent to the paraboloid $\Pi : z = -x^2 - y^2$. A line $p^* \cap q^*$ that is disjoint from Π is contained in exactly two such tangent planes, which determine p and q.

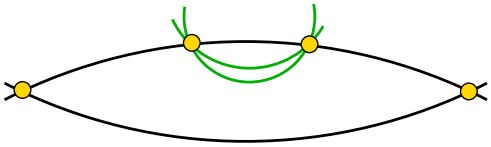


Figure 1 The non-overlapping property may exclude some arcs from participating in a lens.

As a first attempt to bound $|I_{\text{nov}}|$, observe that the Szemerédi-Trotter theorem [13] implies that $|I_{\text{nov}}| = O(n^{2/3}|\Lambda|^{2/3} + n + |\Lambda|)$, and thus, since each $\lambda \in \Lambda$ is k-rich, we have, when k is sufficiently large, $|\Lambda| = O(n^2/k^3 + n/k)$. Unfortunately this bound is too weak to prove Theorem 1. To strengthen the bound, we use a crucial property about non-overlapping lenses, which implies that few of the incidences in I_{nov} can concentrate in a plane.

▶ **Lemma 5.** *Let C be a set of circles and let Λ be a set of pairwise non-overlapping lenses in C. Let $\lambda_1, \lambda_2, \lambda_3 \in \Lambda$ be distinct lenses, and suppose that there is a circle $c \in C$ that participates in all three lenses, that is, $(c^*, \lambda_i^*) \in I_{\text{nov}}$ for $i = 1, 2, 3$. Then $\lambda_1^*, \lambda_2^*, \lambda_3^*$ are not coplanar.*

Proof. Throughout the paper, $|ab|$ denotes the length of the segment ab (the Euclidean distance between a and b). The transformations $c \mapsto c^*$ and $\lambda \mapsto \lambda^*$ described above have the following property. For each nonvertical plane $h \subset \mathbb{R}^3$ there exists a point $w \in \mathbb{R}^2$ and a *power*[4] π such that

$$h = \{c^* \mid c \text{ is a circle, and } w \text{ has power } \pi \text{ with respect to } c\}.$$

Next, suppose that $\lambda_1^*, \lambda_2^*, \lambda_3^*$ lie in a common nonvertical plane h (this plane must necessarily contain c^*), and let w and π be the point and power associated with h. Let p_i and q_i be the vertices of λ_i, for $i = 1, 2, 3$. It is then easy to see that w must lie on each of the lines (in the xy-plane) through p_i and q_i, for $i = 1, 2, 3$, and the power π is $\pm|wp_i| \cdot |wq_i|$, where the sign is positive (resp. negative) if w lies outside (resp. inside) the segment p_iq_i; this is so because any other point cannot have a fixed power with respect to all the circles c whose dual points c^* lie on λ_i^*. In other words, the lines through p_1q_1, p_2q_2, and p_3q_3 are concurrent and meet at w. This however is impossible, because the circle c participates in all three lenses, which implies, as is easily verified (see Figure 2), that at least two of the lenses λ_1, λ_2, λ_3 must be overlapping, a contradiction that completes the proof for nonvertical planes.

The situation is similar when h is vertical. In this case all the circles c for which $c^* \in h$ are centered at points on the line ℓ of intersection of h with the xy-plane. For a lens $\lambda = \lambda_{p,q}$, the associated line λ^* is contained in h if and only if ℓ is the bisector of pq. Again, the fact that the lenses of Λ are pairwise non-overlapping is easily seen to imply that a circle c can contain at most two pairs p, q such that $\lambda_{p,q}$ is a lens in L, with $\lambda^* \subset h$, and c participates in $\lambda_{p,q}$. Hence, $\lambda_1^*, \lambda_2^*, \lambda_3^*$ cannot all lie in h. ◀

Note that a point c^* can be incident to arbitrarily many lines on a plane h, but Lemma 5 implies that at most two of them can contribute to I_{nov}.

▶ **Corollary 6.** *Let C be a set of circles and let Λ be a set of pairwise non-overlapping lenses in C. Let $h \subset \mathbb{R}^3$ be a plane, and let $C' \subset C^*$ and $L' \subset \Lambda^*$ be the set of points and lines contained in h, respectively. Then*

$$|I_{\text{nov}}(\Lambda) \cap I(C',\ L')| \leq 2|C'|.$$

[4] Recall that the power of a point w with respect to a circle c, centered at ξ and having radius r, is $|w\xi|^2 - r^2$.

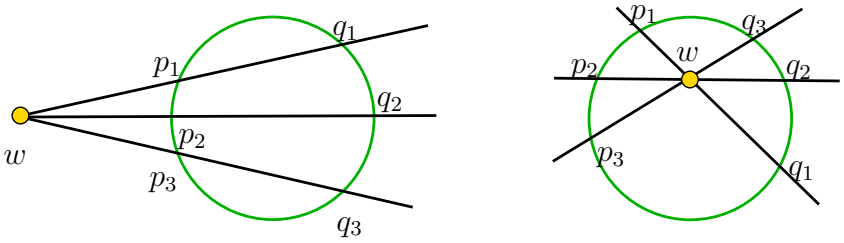


Figure 2 The lines in $\mathbb{R}^3$ corresponding to three pairwise non-overlapping lenses that share a common circle, which participates in all three of them, cannot be coplanar. (They are not the lines in the xy-plane drawn in the figure.)

Corollary 6 suggests that lines contained in a plane contribute few incidences to I_{nov}. Later in our arguments, we will need to study the contribution to I_{nov} coming from lines contained in an algebraic surface. The following lemma says that after removing a small number of ill-behaved lines, the contribution to I_{nov} is still small.

▶ **Lemma 7.** *Let C be a set of circles and let Λ be a set of pairwise non-overlapping lenses in C. Let $P \in \mathbb{R}[x,y,z]$ be a polynomial of degree D, and let $L' \subset \Lambda^*$ be a set of lines contained in the zero set $Z(P)$ of P. Then there is a set $L'' \subset L'$ of cardinality at most $11D^2$, so that*

$$|I_{\text{nov}}(\Lambda) \cap I(C^* \cap Z(P), L' \backslash L'')| \leq 2|C^* \cap Z(P)| + D|L' \backslash L''|.$$

Proof. This result follows immediately from the statements in Guth and Katz [7, Section 3], so we just briefly sketch the proof. Write $Z(P) = Z^{(1)} \cup Z^{(2)} \cup Z^{(3)} \cup Z^{(4)}$, where $Z^{(1)}$ is a union of planes, $Z^{(2)}$ is a union of reguli, $Z^{(3)}$ is a union of irreducible surfaces that are singly ruled by lines and are not planes or reguli, and $Z^{(4)}$ is the union of all irreducible components of $Z(P)$ that are not ruled. Let L'' be the set of lines contained in $Z^{(4)}$. By [7, Corollary 3.3] we have $|L''| \leq 11D^2$.

Let $Z_1, \ldots, Z_h$ be the irreducible components of $Z(P)$. For each index $i = 1, \ldots, h$, let C_i^* be the set of points $c^* \in C^*$ that are contained in Z_i and are not contained in any Z_j with $j < i$. Similarly, let L_i be the set of lines $\ell \in L' \backslash L''$ that are contained in Z_i and are not contained in any Z_j with $j < i$ (note that if the component Z_i is not ruled, then by definition L_i is empty).

First, we count the number of incidences $(c^*, \ell) \in I(C^* \cap Z(P), L' \backslash L'')$ for which $c^* \in C_i$ and $\ell \in L_j$ with $j \neq i$. For such an incidence, we must have that ℓ properly intersects Z_i. Thus there are at most $(D-1)|L' \backslash L''|$ incidences of this form.

Next we count the number of incidences $(c^*, \ell) \in I(C^* \cap Z(P), L' \backslash L'')$ for which $c^* \in C_i$ and $\ell \in L_i$. If Z_i is a plane, then by Corollary 6, there are $\leq 2|C_i^*|$ incidences of this type. If Z_i is a regulus, and hence doubly ruled, then it immediately follows that there are at most $2|C_i^*|$ incidences of this type. Finally, if Z_i is singly ruled, then Z_i has at most one exceptional point (incident to infinitely many lines contained in Z_i), and at most two exceptional (non-generator) lines, in the terminology of [7], which then implies that there are at most $2|C_i^*| + |L_i|$ incidences of this type. Summing the above contributions, we conclude

$$\begin{aligned} |I_{\text{nov}} \cap I(C^* \cap Z(P), L' \backslash L'')| &\leq (D-1)|L' \backslash L''| + \sum_i (2|C_i^*| + |L_i|) \\ &= 2|C^* \cap Z(P)| + D|L' \backslash L''|. \end{aligned}$$

◀

▶ **Proposition 8.** *Let C be a set of circles and let Λ be a set of pairwise non-overlapping lenses in C. Then there is an absolute constant A so that*

$$|I_{\text{nov}}(\Lambda)| \leq A\big(|C|^{1/2}|\Lambda|^{3/4} + |C| + |\Lambda|\big). \tag{1}$$

Proof. The proposition is a slight variant of Guth and Katz's point-line incidence bound from [7], so we just briefly sketch the proof. We prove the result by induction on $|\Lambda|$. Let $M = |C|$, let $L = \Lambda^*$, and let $N = |L|$. First we can suppose that $N \leq M^2$. If not, then Proposition 8 follows immediately from the Kővári-Sós-Turán theorem (see [2, Theorem 9.5]), because the incidence graph of the points and the lines does not contain $K_{2,2}$ as a subgraph.

Let $D = \lfloor \min\left\{M^{1/2}N^{-1/4},\ N^{1/2}/10\right\}\rfloor$. We can suppose that $N^{1/2}/10$ (and thus D) is at least one, since otherwise $|I_{\text{nov}}(\Lambda)| \leq 100|C|$ and we are done. Using the polynomial partitioning for varieties established by Guth [6], we can find a polynomial $P \in \mathbb{R}[x, y, z]$ of degree $\leq D$ so that $\mathbb{R}^3 \backslash Z(P)$ is a union of $O(D^3)$ open connected sets (such sets are often called *cells*), so that each cell contains $O(M/D^3)$ points from C^*, and each cell is intersected by $O(N/D^2)$ lines from L. If $D = N^{1/2}/10$ then each cell intersects $O(1)$ lines from L. Since each point from C^* is contained in at most one cell, we have in this case

$$I(C^* \backslash Z(P), \Lambda) = O(M).$$

If $D = M^{1/2}N^{-1/4}$, then standard incidence estimates allow us to bound

$$I(C^* \backslash Z(P), \Lambda) = O(M^{1/2}N^{3/4}).$$

Similarly, standard incidence estimates allow us to bound

$$|\{(c^*, \ell) \in I(C^* \cap Z(P), \Lambda) \mid \ell \not\subset Z(P)\}| = O(ND) = O(M^{1/2}N^{3/4}).$$

Let $L' \subset L$ be the set of lines contained in $Z(P)$. Applying Lemma 7, we obtain a set $L'' \subset L'$ with $|L''| \leq 11D^2 \leq |L|/2$, and

$$|I_{\text{nov}} \cap I(C^* \cap Z(P), L' \backslash L'')| \leq 2|C^* \cap Z(P)| + D|L' \backslash L''| = O(M^{1/2}N^{3/4} + M).$$

Finally, we apply the induction hypothesis to bound

$$|I_{\text{nov}} \cap I(C^* \cap Z(P), L'')| \leq A\big(M^{1/2}|L''|^{3/4} + M + |L''|\big) \leq 2^{-3/4}AM^{1/2}N^{3/4} + A(M+N).$$

Combining these bounds, we conclude that

$$|I_{\text{nov}}| \leq 2^{-3/4}AM^{1/2}N^{3/4} + A(M+N) + O(M^{1/2}N^{3/4}),$$

with implicit constant independent of A. Selecting A sufficiently large closes the induction. ◀

If C is a set of n circles and Λ is a set of pairwise non-overlapping k-rich lenses in C, then $|I_{\text{nov}}(\Lambda)| \geq k|\Lambda|$. Substituting this inequality in (1), we see that if n is sufficiently large ($n > 8A^3$ will suffice), then $|\Lambda| = O\left(\frac{n^2}{k^4} + \frac{n}{k}\right)$. Thus for all values of $n = |C|$, we have

$$\deg(\Lambda) = |I_{\text{nov}}(\Lambda)| = O\left(\frac{n^2}{k^3} + n\right). \tag{2}$$

In particular, Theorem 1 is true when $k \geq n^{1/3}\log^{-2/3} n$.

In the next two sections, we will prove Theorem 1 when $2 < k < n^{1/3}$, and also give a second proof of a slightly weaker bound. Note that Theorem 1 consists of two statements: a bound on $|\Lambda|$ and a bound on $\deg(\Lambda)$. The second statement immediately implies the first, by dividing the resulting bound by k. The next lemma shows that the first statement also implies the second.

▶ **Lemma 9.** *Suppose that for every set C of circles in the plane and every $k \geq 2$, every set of pairwise disjoint k-rich lenses in C has cardinality at most $A\left(\frac{|C|^{3/2}\log(|C|/k^3)}{k^{5/2}} + \frac{|C|}{k}\right)$. Then for every set C of circles in the plane, every $k \geq 2$, and every set Λ of pairwise disjoint k-rich lenses in C, we have $\deg(\Lambda) = O\left(\frac{|C|^{3/2}\log(|C|/k^3)}{k^{3/2}} + |C|\right)$, where the implicit constant depends only on A.*

Proof. Let C be a set of n circles in the plane. Let $k \geq 2$ and let Λ be a set of pairwise disjoint k-rich lenses in C. If $k \geq n^{1/3}$, then by (2) we have $\deg(\Lambda) = O(n)$ and we are done.

Suppose now that $2 \leq k \leq n^{1/3}$. Let $\Lambda_0 \subset \Lambda$ be the set of lenses that are $n^{1/3}$-rich. Let j_0 be the smallest integer so that $2^{-j_0}n^{1/3} \leq k$, and for each $j = 1, \ldots, j_0$, let $\Lambda_j \subset \Lambda \setminus \bigcup_{i=0}^{j-1}\Lambda_j$ be the set of lenses that are $2^{-j}n^{1/3}$-rich. By construction, $\Lambda = \bigsqcup_{j=0}^{j_0}\Lambda_j$, and for each index $1 \leq j \leq j_0$, the lenses in Λ_j have degree between $2^{-j}n^{1/3}$ and $2^{-j+1}n^{1/3}$. Thus

$$\begin{aligned}
\deg(\Lambda) &= \deg(\Lambda_0) + \sum_{j=1}^{j_0}\deg(\Lambda_j) \\
&\leq \deg(\Lambda_0) + \sum_{j=1}^{j_0}(2^{-j+1}n^{1/3})|\Lambda_j| \\
&\leq O(n) + \sum_{j=1}^{j_0}(2^{-j+1}n^{1/3})\left(\frac{An^{3/2}\log 2^{3j}}{(2^{-j}n^{1/3})^{5/2}}\right) \\
&= O(n) + O\left(2^{\frac{3}{2}j_0}\cdot\frac{n^{3/2}\log 2^{3j_0}}{n^{1/2}}\right) \\
&= O\left(\frac{n^{3/2}\log(n/k^3)}{k^{3/2}} + n\right),
\end{aligned}$$

where the implicit constant depends on A. ◀

▶ Remark 10. Recall that at the beginning of this section, we added the assumption that no two distinct lenses in Λ share the same pair $\{p, q\}$ of endpoints. We can now explain why this assumption is harmless. Indeed, let C be a set of circles and let Λ be a set of pairwise non-overlapping k-rich lenses in C. Let Λ' be the set of lenses formed by "merging" all lenses in Λ that share common endpoints, i.e., if $\lambda_{p,q}(C')$ and $\lambda_{p,q}(C'')$ are k-rich lenses in Λ, then $\lambda_{p,q}(C' \sqcup C'')$ will replace these two lenses in Λ'. While $|\Lambda'|$ might be smaller than $|\Lambda|$, we have $\deg(\Lambda') = \deg(\Lambda)$, because Λ consists of pairwise non-overlapping lenses. To summarize: if we can prove that every set of k-rich lenses with distinct pairs of endpoints has cardinality $O\left(\frac{|C|^{3/2}\log(|C|/k^3)}{k^{5/2}} + \frac{|C|}{k}\right)$, then this implies that every set Λ' of k-rich lenses with distinct endpoints has degree $\deg(\Lambda') = O\left(\frac{|C|^{3/2}\log(|C|/k^3)}{k^{3/2}} + |C|\right)$. This implies the same bound for any set Λ of k-rich lenses (i.e., the distinct endpoint requirement can be dropped).

3 First Proof of Theorem 1: Reduction to Small k

Let C be a set of n circles in the plane, and let Λ be a set of pairwise non-overlapping k-rich lenses in C. Let C^*, $L = \Lambda^*$, and I_{nov} be as defined in Section 2, and let $2 \leq k \leq n^{1/3}$. By Lemma 9, to prove Theorem 1 it suffices to show that $|\Lambda| = O\left(\frac{n^{3/2}\log(n/k^3)}{k^{3/2}}\right)$. Let $\alpha > 0$ be a small absolute constant that will be specified below. We will suppose that $2/\alpha \leq k \leq \frac{1}{10}n^{1/3}$, since otherwise Theorem 1 follows from (2).

We can assume that $|\Lambda| \geq 16n/k$, since otherwise we are done. Together with the inequality $k \leq n^{1/3}/10$, this implies that $|\Lambda| \geq 100k^2$ (the constant is actually larger, but 100 will suffice for our purpose). Let $D = \alpha k$. As in [7], we construct a partitioning polynomial f of degree $O(D)$, so that each of the $O(D^3)$ cells of $\mathbb{R}^3 \setminus Z(f)$ contains at most n/D^3 points of C^* (note that some points of C^* might lie on the zero set $Z(f)$).

Let $L' \subset L$ be the set of lines contained in $Z(f)$. By Lemma 7, there is a set $L'' \subset L'$ with $|L''| \leq 11 \deg(f)^2 = O(\alpha^2 k^2)$ so that

$$|I_{\text{nov}} \cap I(C^* \cap Z(P), L' \setminus L'')| \leq 2|C^* \cap Z(P)| + D|L' \setminus L''| \leq 2n + \alpha k |L' \setminus L''|.$$

Recall that $|L| = |\Lambda| \geq 100k^2$, and thus if $\alpha > 0$ is chosen sufficiently small then $|L''| \leq |L|/4$. Since each line in $L' \setminus L''$ participates in at least k incidences in I_{nov}, we have

$$|L' \setminus L''| \leq \frac{1}{k}(2n + \alpha k |L' \setminus L''|) \leq \frac{2n}{k} + \alpha |L| \leq \frac{|L|}{4},$$

where the final inequality follows by choosing $\alpha < 1/8$ and by using the assumption that $|\Lambda| \geq 16n/k$. We conclude that $|L \setminus L'| \geq |L|/2$. Next, each $\ell \in L \setminus L'$ participates in at least k incidences in I_{nov}, at least $k - \deg(f) \geq (1 - O(\alpha))k$ of which must be inside the cells of $\mathbb{R}^3 \setminus Z(f)$. We say an incidence $(c^*, \ell) \in I_{\text{nov}}$ is *lonely* if c^* is inside a cell of $\mathbb{R}^3 \setminus Z(f)$, and (c^*, ℓ) is the only incidence in I_{nov} involving ℓ that occurs inside that cell (i.e., there are no other points of C^* on ℓ inside that cell, so in the primal plane this implies that this configuration does not form a lens). Since each $\ell \in L \setminus L'$ intersects at most $\deg(f)+1 \leq \alpha k + 1$ cells, each $\ell \in L \setminus L'$ participates in at least $(1 - \alpha)k - 1$ incidences in I_{nov} that are not lonely. Let I'_{nov} be the set of incidences $(c^*, \ell) \in I_{\text{nov}}$ where c^* is inside a cell of $\mathbb{R}^3 \setminus Z(f)$, and the incidence is not lonely. Then if $\alpha > 0$ is selected sufficiently small, we have

$$|I'_{\text{nov}}| \geq |L \setminus L'|(k/2) \geq \frac{1}{4} k |L|. \tag{3}$$

On the other hand, Theorem 4 says that there are $O\big((n/D^3)^{3/2} \log(n/D^3)\big)$ non-lonely incidences inside each cell. Thus

$$|I'_{\text{nov}}| = O\left(D^3 \left(\frac{n}{D^3}\right)^{3/2} \log\left(\frac{n}{D^3}\right)\right) = O\left(\frac{n^{3/2} \log(n/k^3)}{k^{3/2}}\right), \tag{4}$$

where the implicit constant depends on α. Combining (3) and (4), we conclude that

$$|L| = O\left(\frac{n^{3/2} \log(n/k^3)}{k^{5/2}}\right).$$

This completes the first proof of Theorem 1.

▶ Remark 11. It is an interesting challenge to extend the analysis in this section from circles to more general families of algebraic curves. This topic will be discussed in Section 6.

4 Second Proof of Theorem 1: Reduction to Large k

In this section we prove a slightly weaker version of Theorem 1 using a different proof technique. We feel that each of the techniques is interesting in its own right, and that each has the potential of being extended into different and more general contexts.

Most of the analysis in this section extends to more general algebraic curves, except for one (significant) step. We will discuss possible generalizations in Section 6.

Sharpening our notation from the previous sections, we define $F(n,k)$ to be the smallest integer with the following property: Let C be a set of at most n circles in the plane; let Λ be a set of pairwise disjoint k-rich lenses in C. Then $\deg(\Lambda) \leq F(n,k)$. Note that $F(n,1) = \infty$ (i.e., it is undefined for $k = 1$), and, trivially, $F(n,k) = O(n^2)$ for all $k \geq 2$. Furthermore, $F(n,k)$ is monotone increasing in n and monotone decreasing in k. Abusing notation slightly, we extend our definition of $F(n,k)$ to all real numbers $n \geq 1$ and $k \geq 2$ by defining $F(n,k) = F(\lfloor n \rfloor, \lceil k \rceil)$. Finally, note that if A and n are integers, then $F(An,k) \geq AF(n,k)$, since we may take A disjoint copies of a configuration of n circles that achieves the $F(n,k)$ bound.

In Section 2 we proved that $F(n,k) = O(n^2/k^3 + n)$, and in particular there is an absolute constant A_0 so that, for any $z > 1$ and any k,

$$F(k^3 z, k) \leq A_0 k^3 z^2. \tag{5}$$

In this section we will establish the following recurrence relation for $F(n,k)$.

▶ **Lemma 12.** *There is a constant A so that for any $D, n, k \geq 1$ we have*

$$F(n,k) \leq AD^3 F(n/D^2, k/4) + F(AD^2, k/4) + AD^2 n. \tag{6}$$

Before proving Lemma 12, we show that it implies

$$F(n,k) = O\left(\frac{n^{3/2}\log^b(n/k^3)}{k^{3/2}}\right), \tag{7}$$

for some constant b and for all $n > k^3$. To show this, we solve the recurrence in the lemma in several steps. First, given n and k, we construct a sequence of real numbers $n_0, n_1, \ldots, n_s = n$, where $n_0 = k^3 z$, for a suitable value of $z > 1$ (the actual value will be between $\sqrt{2}$ and 2, and its concrete choice will be given towards the end of the forthcoming analysis), and $n_{j+1} = n_j^2/k^3$, for $j \geq 0$. That is, $n_j = k^3 z^{2^j}$ for $j \geq 0$, as is easily verified by induction on j. Since we want n_s to be equal to n, we have $z^{2^s} = n/k^3$. We also define, for each j, $D_j := n_j^{1/2}/k^{3/2} = z^{2^{j-1}}$, and note that $n_{j+1} = D_j^2 n_j$. The rationale for choosing these sequences will become clear as the solution of the recurrence unfolds. We have

$$\frac{n_{j+1}^{3/2}}{k^{3/2}} = \frac{D_j^3 n_j^{3/2}}{k^{3/2}} = \frac{D_j^2 n_j^2}{k^3} = D_j^2 n_{j+1}. \tag{8}$$

Note that $AD_j^2 = An_j/k^3$. For simplicity, we shall suppose that $k \geq A^{1/3}$ and thus $AD_j^2 \leq n_j$ (if this inequality failed then Theorem 1 follows from Theorem 4, since k becomes a constant).

We next prove that for each $j \geq 0$ we have

$$F(n_j, 4^j k) \leq A_0 z^{1/2} (4A)^j n_j^{3/2}/k^{3/2}, \tag{9}$$

where A_0 is the constant from (5). The case $j = 0$ is precisely (5). For the induction step, we compute, using Lemma 12:

$$\begin{aligned}
F(n_{j+1}, 4^{j+1}k) &\leq AD_j^3 F(n_{j+1}/D_j^2, 4^j k) + F(AD_j^2, 4^j k) + AD_j^2 n_{j+1} \\
&\leq 2AD_j^3 F(n_j, 4^j k) + AD_j^2 n_{j+1} \\
&\leq 2AD_j^3 (A_0 z^{1/2})(4A)^j n_j^{3/2}/k^{3/2} + AD_j^2 n_{j+1} \\
&\leq 2 \cdot 4^j (A_0 z^{1/2}) A^{j+1} n_{j+1}^{3/2}/k^{3/2} + AD_j^2 n_{j+1} \\
&\leq A_0 z^{1/2} (4A)^{j+1} n_{j+1}^{3/2}/k^{3/2},
\end{aligned}$$

where in the last inequality we used the equality (8), namely $D_j^2 n_{j+1} = n_{j+1}^{3/2}/k^{3/2}$.

Thus if $n > k^3$, we can find $z > 1$ and s so that $n_s = k^3 z^{2^s} = n$ i.e., $2^s = \frac{\log(n/k^3)}{\log z}$ or $s = \log\log(n/k^3) - \log\log z$. Using (9), with $j = s$, $n_j = n$ and replacing k by $k/4^s$ we get

$$F(n,k) \leq A_0 z^{1/2} (2^5 A)^s \frac{n^{3/2}}{k^{3/2}}.$$

Putting $B := 2^5 A$ and $b := \log B$, we get

$$(2^5 A)^s = B^s = (2^s)^b = \left(\frac{\log(n/k^3)}{\log z}\right)^b = \frac{\log^b(n/k^3)}{\log^b z},$$

and hence

$$F(n,k) \leq A_0 z^{1/2} B^s \frac{n^{3/2}}{k^{3/2}} = \frac{A_0 z^{1/2}}{\log^b z} \cdot \frac{n^{3/2}\log^b(n/k^3)}{k^{3/2}}.$$

It remains to determine the value of z. Put $z_j = (n/k^3)^{1/2^j}$, for $j \geq 0$. This sequence converges to 1 and satisfies $z_j = \sqrt{z_{j-1}}$ for each j. We take s to be that (unique) value of j for which $\sqrt{2} < z_j \leq 2$ (for such a z to exist we need to assume that $n > k^3\sqrt{2}$). We then have

$$\frac{A_0 z^{1/2}}{\log^b z} \leq A_1 := \frac{A_0\sqrt{2}}{\log^b \sqrt{2}}, \qquad \text{and so} \qquad F(n,k) \leq A_1 \frac{n^{3/2}\log^b(n/k^3)}{k^{3/2}}.$$

This establishes (7), and leaves us with the task of proving Lemma 12.

Proof of Lemma 12. Let C be a set of n circles in the plane and let Λ be a set of pairwise non-overlapping k-rich lenses in C. Following the technique of Ellenberg, Solymosi, and Zahl [5], for each circle $c \in C$ with defining polynomial g (i.e., $c = Z(g)$), consider the variety

$$\{(x,y,z) \in \mathbb{R}^3 \mid g(x,y) = 0,\ z\partial_y g(x,y) + \partial_x g(x,y) = 0\}.$$

As discussed in [5, Section 3.3], this variety is a union of three irreducible curves in $\mathbb{R}^3$, two of which are vertical lines (one above each of the points in c where the circle has infinite slope). Define $\gamma(c) \subset \mathbb{R}^3$ to be the irreducible component that is not a vertical line. If $(x,y) \in c$ is a point where c has finite slope, then $(x,y,z) \in \gamma(c)$ if and only if c has slope z at (x,y). In particular, if $\lambda_{p,q}(C')$ is a lens and if $c \in C'$, then the (shorter) arc $\beta \subset c$ with endpoints p and q lifts to a curve segment $\gamma(\beta) \subset \gamma(c)$. We will call this curve segment the *lifted arc* of c corresponding to the lens λ.

For a set of circles C, define $\gamma(C) = \{\gamma(c) \mid c \in C\}$. Let $\Lambda_{p,q}(C')$ be a lens in C, and suppose that none of the circles $c \in C'$ have infinite slope at the point p or q (this is a harmless assumption, since at most two circles containing p and q can have infinite slope at p or q). Let $\ell_p, \ell_q \subset \mathbb{R}^3$ be vertical lines passing through $(p,0)$ and $(q,0)$ respectively. Then each of the curves in $\gamma(C')$ intersect ℓ_p and ℓ_q. Furthermore, each of the intersection points $\{\gamma(c) \cap \ell_p \mid c \in C'\}$ are distinct, and similarly for ℓ_q. Define $z(\gamma(c) \cap \ell_p)$ to be the z-coordinate of $\gamma(c) \cap \ell_p$. The curves in $\gamma(C')$ have the following property:

Order Reversal Property. If we order the curves $c_1, \ldots, c_m \in C'$ so that

$$\begin{aligned} &z(\gamma(c_1) \cap \ell_p) < z(\gamma(c_2) \cap \ell_p) < \cdots < z(\gamma(c_m) \cap \ell_p), \\ \text{then}\ \ &z(\gamma(c_1) \cap \ell_q) > z(\gamma(c_2) \cap \ell_q) > \cdots > z(\gamma(c_m) \cap \ell_q). \end{aligned}$$

I.e., the order on C' given by the z-coordinates of $\gamma(c) \cap \ell_p$ is precisely the reverse of the order given by $\gamma(c) \cap \ell_q$. See Figure 3.

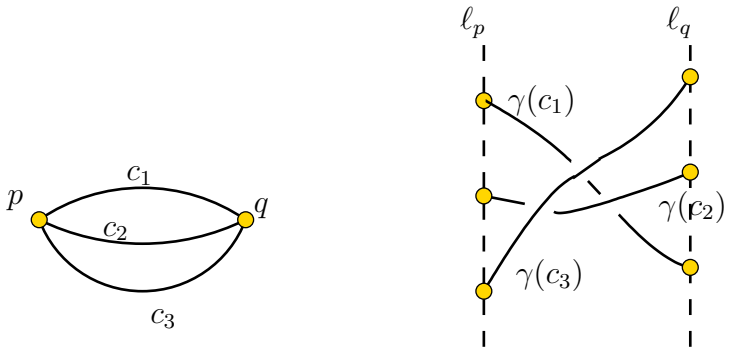


Figure 3 A lens is lifted to a multi-2-cycle in three dimensions.

We employ the approach of Aronov and Sharir [4], as detailed in Sharir and Zahl [10], with some modifications, as follows. We construct a partitioning polynomial f, of degree $O(D)$, so that we have $O(D^3)$ open connected cells of $\mathbb{R}^3 \setminus Z(f)$ (recall that $Z(f)$ is the zero set of f), and at most n/D^2 curves from $\gamma(C)$ intersect each cell. The existence of such a partitioning polynomial was established in Guth [6]. For each cell O of $\mathbb{R}^3 \setminus Z(f)$, define $C_O = \{c \in C \mid \gamma(c) \cap O \neq \emptyset\}$.

For a cell $O \subset \mathbb{R}^3 \setminus Z(f)$, we say that a lens $\lambda = \lambda_{p,q}(C') \in \Lambda$ is *preserved* within O if for at least $\deg(\lambda)/4 = |C'|/4$ circles $c \in C'$, the lifted arc of c corresponding to the lens λ is fully contained in O. In particular, if $\lambda_{p,q}(C')$ is preserved within O, then $|C' \cap C_O| \geq |C'|/4 \geq k/4$. Thus for each cell O, we have

$$\sum \deg(\lambda_{p,q}(C')) \leq 4 \sum |C' \cap C_O| \leq 4F(n/D^2, k/4),$$

where the sum is taken over all lenses $\lambda_{p,q}(C')$ that are preserved within O. Summing over all cells O, we conclude that

$$\sum_{\lambda \text{ preserved within a cell}} \deg(\lambda) = O(D^3)F(n/D^2, k/4). \tag{10}$$

If a lens is not preserved within any cell, we say that it is *disrupted* by $Z(f)$. It remains to bound the sum of the degrees of the disrupted lenses. The arguments here are very similar to those in [10], so we just sketch them briefly. First, for each $(x, y, z) \in \mathbb{R}^3$, define $h(x, y, z)$ to be the number of intersections between $Z(f)$ and the infinite ray $\{(x, y, t) \mid t > z\}$. This quantity is finite (indeed bounded by $\deg f$) unless the vertical line passing through (x, y, z) is contained in $Z(f)$. Following the arguments in [4, 10], there is a polynomial $g \in \mathbb{R}[x, y, z]$ of degree $O(D^2)$ with the following properties.

- h is constant on each connected component of $\mathbb{R}^3 \setminus \big(Z(f) \cup Z(g)\big)$.
- h is constant on $Z(f) \setminus Z(g)$.
- $g(x, y, z)$ is independent of z, i.e., $g(x, y, z) = \tilde{g}(x, y)$ for some polynomial $\tilde{g}(x, y) \in \mathbb{R}[x, y]$.
- If $Q(x, y, z)$ is an irreducible component of f that is independent of z, then Q divides g, i.e., Q is also an irreducible component of g.

In brief, the polynomial $g(x, y, z) = \tilde{g}(x, y)$ is constructed by computing the resultant of f and $\partial_z f$; see [4, 10] for details.

We say that a lens $\lambda_{p,q}(C')$ is *preserved* by $Z(g)$ if at least $\deg(\lambda)/4 = |C'|/4$ of the curves from $\gamma(C')$ are contained in $Z(g)$. Recall that $g(x,y,z) = \tilde{g}(x,y)$, and thus if $\gamma(c) \subset Z(g)$, we must have $c \subset Z(\tilde{g})$. In particular, at most $\deg(g) = \deg(\tilde{g}) = O(D^2)$ circles from C can be contained in $Z(g)$. Arguing as before, we conclude that

$$\sum_{\lambda \text{ preserved by } Z(g)} \deg(\lambda) \leq 4F(O(D^2), k/4) = F(O(D^2), k/4). \tag{11}$$

It remains to bound the sum of the degrees of the lenses that are disrupted by $Z(f)$ and not preserved by $Z(g)$.

Claim. If $\lambda_{p,q}(C')$ is such a lens, then there are at least $|C'|/4 - 1$ circles $c \in C'$ so that the lifted arc of c corresponding to the lens λ properly intersects $Z(f)$ or $Z(g)$.

Once this claim has been established we are done, since the number of such proper intersections is at most $n(\deg f + \deg g) = O(D^2 n)$, and since the lenses are pairwise non-overlapping, each such intersection is counted towards at most one lens.

To verify this claim, let $\lambda_{p,q}(C')$ be a lens that is disrupted by $Z(f)$ and not preserved by $Z(g)$. We will divide our argument into the following two cases.

Case 1. At least half of the lifted circles in C' are contained in $Z(f) \cup Z(g)$.
Since the lens $\lambda_{p,q}(C')$ is not preserved by $Z(G)$, fewer than $|C'|/4$ circles from C' can be contained in $Z(g)$. Thus at least $|C'|/4$ circles from C' are contained in $Z(f)$. Enumerate these circles as $c_1, \ldots, c_w$, for some $w \geq |C'|/4$, so that $z(\gamma(c_1) \cap \ell_p) < \cdots < z(\gamma(c_w) \cap \ell_p)$. Since each circle c_i is contained in $Z(f)$ but not contained in $Z(g)$, we have

$$h(\gamma(c_1) \cap \ell_p) < \cdots < h(\gamma(c_w) \cap \ell_p).$$

However, by the Order Reversal Property, we have $z(\gamma(c_1) \cap \ell_q) > \cdots > z(\gamma(c_w) \cap \ell_q)$, so

$$h(\gamma(c_1) \cap \ell_q) > \cdots > h(\gamma(c_w) \cap \ell_q).$$

Since h is constant on $Z(f) \backslash Z(g)$, we conclude that for all but at most one index i, the lifted arc of c_i corresponding to the lens λ intersects $Z(g)$. Thus for at least $|C'|/4 - 1$ circles $c \in C'$, the lifted arc of c corresponding to the lens λ properly intersects $Z(g)$.

Case 2. At least half of the lifted circles in C' are not contained in $Z(f) \cup Z(g)$.
Let $c_1, \ldots, c_w$, for some $w \geq |C'|/2$, be circles in C' whose lifted curve is not contained in $Z(f) \cup Z(g)$. For each index $i = 1, \ldots, w$, let β_i be the (shorter) arc of c_i with endpoints p and q. Let v be the number of arcs $\gamma(\beta_i)$ that properly intersect $Z(f)$; if $v \geq |C'|/4$ then we are done. If not, then at least $w - v$ of the arcs $\gamma(\beta_i)$ are contained inside a cell of $Z(f)$ (though different arcs might be contained inside different cells). But an argument analogous to the one above shows that all of the arcs in all but one of the cells must properly intersect $Z(g)$. Since no cell contains more than $|C'|/4$ of the lifted arcs, at least $w - v - |C'|/4$ of the lifted arcs must properly intersect $Z(g)$. We conclude that v arcs properly intersect $Z(f)$ and at least $w - v - |C'|/4 \geq |C'|/4 - v$ arcs properly intersect $Z(g)$. Thus at least $|C'|/4$ arcs properly intersect either $Z(f)$ or $Z(g)$.

Combining the bounds in (10), (11), adding the overhead $O(D^2 n)$, and making the constants in the $O(\cdot)$ notation explicit, bounding all of them by the same constant A, we obtain the recurrence asserted in the lemma. ◀

5 Point-Circle and Lens-Circle Incidence Bounds

5.1 Point-circle incidence bounds

We can use Theorem 1 to bound the number of incidences between m points and n circles in the plane. As it turns out, the bound that we get is the same as the best known bound due to Agarwal et al. [1] (and to [8]). We describe the derivation nonetheless, as an illustration of the power of Theorem 1.

Let P be a set of m points and let C be a set of n circles. We fix a parameter k, to be determined below, and use a modified variant of Székely's technique [12]. We first construct a graph G whose vertices are the points of P, and whose edges connect pairs of consecutive points along each circle of C. Some edges of G form k-rich lenses, and we observe that these lenses are pairwise non-overlapping. Let Λ denote the set of these lenses. We split G into two subgraphs G_0 and G_1, where G_1 consists of all the edges in the lenses of Λ and G_0 consists of all the remaining edges.

By Theorem 1, the number of edges of G_1 is $O\left(\frac{n^{3/2}\log(n/k^3)}{k^{3/2}} + n\right)$.

The number $E_0(c)$ of edges of G_0 along a circle c is $|N_c| - E_1(c)$, where $N_c = P \cap c$ and $E_1(c)$ is the number of edges of G_1 along c. Note that the multiplicity of each edge of G_0 is smaller than k. An upper bound on the number of edges of G_0 then follows from a variant of Székely's technique (see Theorem 8 of [12]), which takes into account the maximum multiplicity of an edge in the graph (which is smaller than k). Concretely, denoting by $|G_0|$ (resp., $|G_1|$) the number of edges of G_0 (resp., G_1), we have

$$|G_0| = O\left(k^{1/3}m^{2/3}n^{2/3} + km\right), \qquad \text{and thus}$$
$$|G| = |G_0| + |G_1| = O\left(\frac{n^{3/2}\log(n/k^3)}{k^{3/2}} + k^{1/3}m^{2/3}n^{2/3} + km + n\right).$$

We balance the first two terms in the bound for $|G|$ by choosing $k = n^{5/11}(\log(n/k^3))^{6/11}/m^{4/11}$. This is meaningful when $k \geq 1$, which holds when $m \leq n^{5/4}\log^{3/2} n$, which is indeed the interesting range. For larger values of m, we take $k = 2$ and get the bound $O(m^{2/3}n^{2/3} + m + n^{3/2}\log n)$, which is dominated by $O(m^{2/3}n^{2/3} + m)$. The bound then becomes (see [1])

$$O\left(m^{2/3}n^{2/3} + m^{6/11}n^{9/11}\log^{2/11}(m^3/n) + m + n\right).$$

Note that the bound is meaningful only for $m > n^{1/3}$. For smaller values of m, the bound becomes $O(n)$. The logarithmic factor provides a "smooth" transition from the above bound to the linear bound as $m \searrow n^{1/3}$.

5.2 Circle-lens incidence bounds

We can apply the bounds in Theorem 1 to obtain an upper bound on the number of incidences between m pairwise non-overlapping lenses and n circles, where a lens λ is said to be incident to a circle c if c participates in λ. To do so, let Λ be the given set of m lenses (which are not necessarily rich). Set $k := \frac{n^{3/5}\log^{2/5}(n/k^3)}{m^{2/5}}$. (Note that $k = \Omega(1)$ since we always have $m = O(n^{3/2}\log n)$ [1, 8].) The non-k-rich lenses of Λ contribute at most $km = m^{3/5}n^{3/5}\log^{2/5}(n/k^3)$ incidences. The k-rich lenses contribute, by Theorem 1,

$$O\left(\frac{n^{3/2}\log(n/k^3)}{k^{3/2}} + n\right) = O\left(m^{3/5}n^{3/5}\log^{2/5}(n/k^3) + n\right)$$

incidences. Since $\log(n/k^3) = O(\log(m^3/n^2))$, we thus obtain:

▶ **Theorem 13.** *Let Λ be a family of m pairwise non-overlapping lenses in an arrangement of n circles in the plane. Then the number of incidences between the lenses of Λ and the circles of C is $O\left(m^{3/5}n^{3/5}\log^{2/5}(m^3/n^2)+n\right)$.*

▶ Remark 14. Aside from the log factor, this bound generalizes the recent result of Sharir and Zlydenko [11] (see also Sharir, Solomon, and Zlydenko [9]) on incidences between so-called directed points and circles. A directed point is a pair (p,u) where p is a point in the plane and u is a direction, and (p,u) is incident to a circle c if $p\in c$ and u is the direction of the tangent to c at p. The bound in [9, 11] is $O(m^{3/5}n^{3/5}+m+n)$ which is similar, albeit slightly sharper, than the bound in Theorem 13. The two setups are indeed related, as a directed point of degree at least two is a limiting case of a lens, and the resulting infinitesimal limit lenses are clearly pairwise non-overlapping. The novelty in Theorem 13 is that lenses are 4-parameterizable, that is, each lens is specified by four real parameters (the coordinates of its vertices p, q), whereas directed points are 3-parameterizable. This makes the analysis in [9, 11] inapplicable to the case of lenses, and yet the bound is more or less preserved.

6 Discussion

Each of the two proofs of the main result, given in Sections 3 and 4, can be extended to more general contexts, provided that certain key properties can be established, or alternatively are assumed. In this section we discuss such possible extensions, and then summarize the state of affairs developed in this paper.

First proof. We offer a few informal comments on a possible approach to extending Theorem 1 to more general plane curves. First, we need to assume that the curves in our family C are 3-parameterizable, so that we can represent them as points in a dual 3-space, and also that they are algebraic of some constant degree. Each point $p\in\mathbb{R}^2$ then becomes a two-dimensional surface p^*, consisting of the points in $\mathbb{R}^3$ whose corresponding curves contain p. Then a lens with endpoints p, q becomes the curve $\ell_{p,q}=p^*\cap q^*$ (we ignore in this informal discussion various issues involving degeneracies and various assumptions that one might need to impose).

We can then apply the same partitioning argument. Inside each cell, we use the (slightly weaker) bound $O(n^{3/2}\text{polylog}(n))$, due to Sharir and Zahl [10], on the number of lenses formed by a set of bounded-degree algebraic curves.

The main difference is in handling points and curves that lie on the zero set of the partitioning polynomial. The preceding analysis strongly relied on Lemma 5, which requires that the curves in C be circles. This in turn allowed us to control the number of incidences occurring on the zero-set $Z(f)$ of the partitioning polynomial. With an analogue of Lemma 5 for more general curves, it seems plausible that the rest of the argument will work with standard modifications.

Second proof. Lemma 12 holds with almost no modification if the circles in C are replaced by arbitrary (bounded degree, algebraic) curves. Indeed, the only important difference is that the Order Reversal Property might not be true, but the dichotomy that a lens must either be preserved within a cell or disrupted by $Z(f)$ remains true, and the bound on the number of lenses that are disrupted by $Z(f)$ and not preserved by $Z(g)$ also remains true. Thus the only obstruction to extending Theorem 1 to more general curves is that the estimate $F(n,n^{1/3})=O(n)$ (or, more precisely, $F(nz,n^{1/3})=O(nz^2)$ for $z>1$), which serves as the base case of the induction, might not be true. We conjecture that for other classes of curves, an estimate of the form $F(n,n^b)=O(n)$ should hold (where $b>0$ depends on the class of curves). As b becomes larger, the corresponding analogue of Theorem 12 becomes weaker.

References

1 P. Agarwal, E. Nevo, J. Pach, R. Pinchasi, M. Sharir, and S. Smorodinsky. Lenses in arrangements of pseudocircles and their applications. *J. ACM*, 51:139–186, 2004.

2 P. Agarwal and J. Pach. *Combinatorial Geometry*. Wiley Interscience, 1995.

3 B. Aronov and M. Sharir. Cutting circles into pseudo-segments and improved bounds for incidences. *Discrete Comput. Geom.*, 28:475–490, 2002.

4 B. Aronov and M. Sharir. Almost tight bounds for eliminating depth cycles in three dimensions. *Discrete Comput. Geom.*, 59:725–741, 2018.

5 J. Ellenberg, J. Solymosi, and J. Zahl. New bounds on curve tangencies and orthogonalities. *Discrete Anal.*, 18, 2016.

6 L. Guth. Polynomial partitioning for a set of varieties. *Math. Proc. Cambridge Philos. Soc.*, 159:459–469, 2015.

7 L. Guth and N.H. Katz. On the Erdős distinct distances problem in the plane. *Ann. of Math.*, 181:155–190, 2015.

8 A. Marcus and G. Tardos. Intersection reverse sequences and geometric applications,. *J. Combin. Theory Ser. A*, 113:675–691, 2006.

9 M. Sharir, N. Solomon, and O. Zlydenko. Incidences with curves with almost two degrees of freedom. *arXiv e-prints*, 2020. `arXiv:2003.02190v2`.

10 M. Sharir and J. Zahl. Cutting algebraic curves into pseudo-segments and applications. *J. Combin. Theory Ser. A*, 150:1–35, 2017.

11 M. Sharir and O. Zlydenko. Incidences between points and curves with almost two degrees of freedom. In *36th International Symposium on Computational Geometry (SoCG 2020)*, volume 164, pages 66:1–66:14, 2020.

12 L. Székely. Crossing numbers and hard Erdős problems in discrete geometry. *Combin. Probab. Comput.*, 11:1–10, 1993.

13 E. Szemerédi and W.T. Trotter. Extremal problems in discrete geometry,. *Combinatorica*, 3:381–392, 1983.

Packing Squares into a Disk with Optimal Worst-Case Density

Sándor P. Fekete
Department of Computer Science, TU Braunschweig, Germany

Vijaykrishna Gurunathan
Department of Computer Science & Engineering, IIT Bombay, India

Kushagra Juneja
Department of Computer Science & Engineering, IIT Bombay, India

Phillip Keldenich
Department of Computer Science, TU Braunschweig, Germany

Linda Kleist
Department of Computer Science, TU Braunschweig, Germany

Christian Scheffer
Department of Computer Science, TU Braunschweig, Germany

Abstract

We provide a tight result for a fundamental problem arising from packing squares into a circular container: The critical density of packing squares into a disk is $\delta = {}^{8}/_{5\pi} \approx 0.509$. This implies that any set of (not necessarily equal) squares of total area $A \leq {}^{8}/_{5}$ can always be packed into a disk with radius 1; in contrast, for any $\varepsilon > 0$ there are sets of squares of total area ${}^{8}/_{5} + \varepsilon$ that cannot be packed, even if squares may be rotated. This settles the last (and arguably, most elusive) case of packing circular or square objects into a circular or square container: The critical densities for squares in a square (${}^{1}/_{2}$), circles in a square (${}^{\pi}/_{(3+2\sqrt{2})} \approx 0.539$) and circles in a circle (${}^{1}/_{2}$) have already been established, making use of recursive subdivisions of a square container into pieces bounded by straight lines, or the ability to use recursive arguments based on similarity of objects and container; neither of these approaches can be applied when packing squares into a circular container. Our proof uses a careful manual analysis, complemented by a computer-assisted part that is based on interval arithmetic. Beyond the basic mathematical importance, our result is also useful as a blackbox lemma for the analysis of recursive packing algorithms. At the same time, our approach showcases the power of a general framework for computer-assisted proofs, based on interval arithmetic.

2012 ACM Subject Classification Theory of computation → Packing and covering problems; Theory of computation → Computational geometry

Keywords and phrases Square packing, packing density, tight worst-case bound, interval arithmetic, approximation

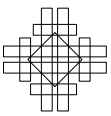
Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.36

Related Version *Full Version*: `https://arxiv.org/abs/2103.07258`

Supplementary Material *Software (Source Code)*:
`https://github.com/phillip-keldenich/squares-in-disk`
archived at `swh:1:dir:7b372039e1b09587f86959e9e1fb013bf2789658`

1 Introduction

Geometric packing and covering problems arise in a wide range of natural applications. They also have a long history of spawning many demanding (and often still unsolved) mathematical challenges. These difficulties are also notable from an algorithmic perspective, as relatively straightforward one-dimensional variants of packing and covering are already NP-hard [16];



however, deciding whether a given set of one-dimensional segments can be packed into a given interval can be checked by computing their total length. This simple criterion is no longer available for two-dimensional, geometric packing or covering problems, for which the total area often does not suffice to decide feasibility of a set, making it necessary to provide an explicit packing or covering. A recent result by Abrahamsen et al. [1] indicates that these difficulties have far-reaching consequences: Two-dimensional packing problems are $\exists\mathbb{R}$-hard, so they are unlikely to even belong to NP.

We provide a provably optimal answer for a natural and previously unsolved case of *tight worst-case area bounds*, based on the notion of *critical packing density*: What is the largest number $\delta \leq 1$, such that any set of squares with a total area of at most δ can always be packed (in a not necessarily axis-parallel fashion) into a disk C of area 1, regardless of the individual sizes of the squares? We show the following theorem that implies $\delta = {}^{8}/_{5\pi} \approx 0.509$ for squares in a disk.

▶ **Theorem 1.** *Every set of squares with a total area of at most* ${}^{8}/_{5}$ *can be packed into the unit disk. This is worst-case optimal, i.e., for every* $A > {}^{8}/_{5}$ *there exists a set of squares with total area* A *that cannot be packed into the unit disk.*

This critical density δ is of mathematical importance, as it settles the last open case of packing circular or square objects into a circular or square container. Figure 1 provides an overview of the critical densities in similar settings, i.e., the critical density for packing squares in a square (${}^{1}/_{2}$), disks in a square (${}^{\pi}/_{(3+2\sqrt{2})} \approx 0.539$), and disks in a disk (${}^{1}/_{2}$).

This result is also of algorithmic interest, because it provides a simple sufficient criterion for feasibility. Note that the previous results illustrated in Figure 1 benefitted from recursive subdivisions of a square container into subpieces bounded by straight lines, or from recursion based on the similarity of objects and container when both are disks; neither applies when objects are squares and the container is a disk. This gives our approach added methodical significance, as it showcases a general framework for establishing computer-assisted proofs for difficult packing problems for which concise manual arguments may be elusive.

A proof of Theorem 1 consists of (i) a class of instances that provide the upper bound of ${}^{8}/_{5\pi}$ for the critical packing density δ and (ii) an algorithm that achieves the matching lower bound for δ by packing any set of squares with a total area of at most ${}^{8}/_{5}$ into the unit

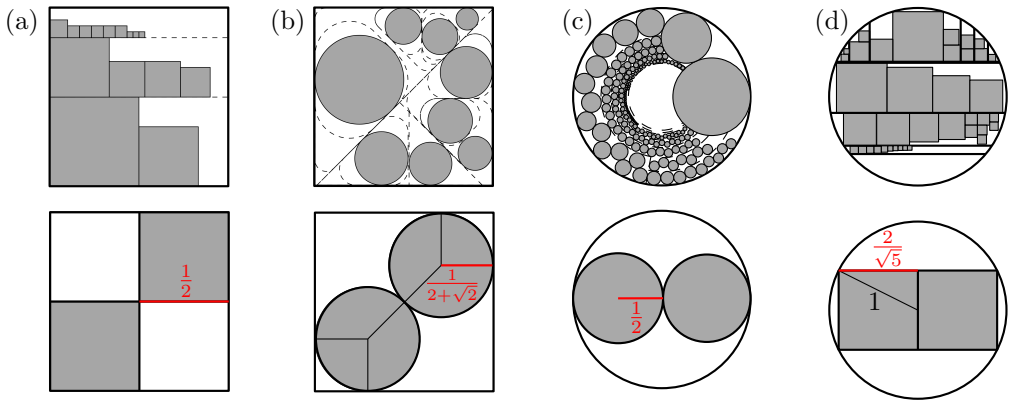


Figure 1 Worst-case optimal approaches and matching worst-case instances for packing: (a) Squares into a square with Shelf Packing by Moon and Moser [23]. (b) Disks into a square by Morr et al. [24, 15]. (c) Disks into a disk by Fekete et al. [14]. (d) Squares into a disk [this paper].

disk. The first part is relatively simple: As shown in Figure 1(d), a critical configuration consists of two squares of side length $s = 2/\sqrt{5}$ and a disk $\mathcal{D}$ of radius 1. It is easy to see that any infinitesimally larger square (of side length $s + \varepsilon$ for any $\varepsilon > 0$) must contain the center of $\mathcal{D}$ in its interior, so two such squares cannot be packed.

The remainder of our paper focuses on the difficult part: providing a strategy for packing sets of squares into a disk (described in Section 2), and then proving that any set of squares with a total area of at most $8/5$ can indeed be packed into the unit disk. This proof is set up with two sets of tools: In Section 3, we describe a general technique that we employed for automated parts of our proof, while Section 4 provides a number of helpful lemmas. Section 5 gives an outline of the actual analysis of our algorithm. Due to space constraints, some detailed proofs are omitted and can be found in the full version of the paper.

1.1 Related work: geometric packing

Problems of geometric packing have been studied for a long time. Providing a survey that does justice to the wide range of relevant work goes beyond the scope of this paper; therefore, we strictly focus on very closely related results, in particular, concerning critical packing density. We refer to Fejes Tóth [10, 28], Lodi, Martello and Monaci [22], Brass, Moser and Pach [8] and Böröczky [7] for more comprehensive surveys.

Even the decision problem whether it is possible to pack a given set of squares into the unit square was shown to be strongly NP-complete by Leung et al. [21], using a reduction from 3-Partition. Already in 1967, Moon and Moser [23] proved that it is possible to pack a set of squares into the unit square if their total area does not exceed $1/2$. This bound is best possible, because two squares even infinitesimally larger than the ones shown in Figure 1(a) cannot be packed. The proof is based on a simple recursive argument.

For the scenario with circular objects, Demaine, Fekete, and Lang [9] showed in 2010 that deciding whether a given set of disks can be packed into a unit square is NP-hard. Using a recursive procedure for partitioning the container into triangular pieces, Morr, Fekete and Scheffer [15, 24] proved that the critical packing density of disks in a square is $\pi/(3+2\sqrt{2}) \approx 0.539$.

More recently, Fekete, Keldenich and Scheffer [14] established the critical packing density of disks into a disk. Employing a number of algorithmic techniques in combination with some interval arithmetic and computer-assisted case checking, they proved that the critical packing density of disks in a disk is $1/2$; they also provide a video including an animated overview [6]. In a similar manner, Fekete et al. [13] established a closed-form description of the total disk area that is sometimes necessary and always sufficient to cover a rectangle depending on its aspect ratio.

Note that the main objective of this line of research is to compute tight worst-case bounds. For specific instances, a packing may still be possible, even if the density is higher; this also implies that proofs of infeasibility for specific instances may be trickier. However, the idea of using the total item volume for computing packing bounds can still be applied. See the work by Fekete and Schepers [11, 12], which shows how a *modified* volume for geometric objects can be computed, yielding good lower bounds for one- or higher-dimensional scenarios.

1.2 Related work: interval arithmetic and computer-assisted proofs

Establishing tight worst-case bounds for packing problems needs to overcome two main difficulties. The first is to deal with the need for accurate computation in the presence of potentially complicated coordinates; the second is the tremendous size of a full case analysis for a complete proof.

Developing methods for both of these challenges has a long tradition in mathematics. One of the first instances of interval arithmetic is Archimedes's classic proof [4] that $^{223}/_{71} \leq \pi \leq {}^{22}/_{7}$, establishing a narrow interval for the fundamental constant of geometry. This entails dealing with inaccurate computation not merely by giving a close approximation, but by establishing an interval for the correct value, which can be used for valid lower and upper bounds for subsequent computations.

Employing electronic devices (e.g., calculators or computer algebra) for mathematical arguments is a well-established method for eliminating tedious, error-prone manual calculations. A famous milestone for the role of computers in theorem proving itself is the confirmation of the Four Color Theorem, a tantalizing open problem for more than 100 years [29]. While the first pair of papers by Appel and Haken [2, 3] was still disputed, the universally accepted proof by Robertson et al. [26] still relies on extensive use of automated checking.

Another example is the resolution of the Kepler conjecture by Hales et al. [17]: While the first version of the proof [18] was still met with some skepticism, the revised and cleaned up variant [17] fits the mold of a more traditional proof, despite relying both on combinatorial results and computational case checking. Note that this proof uses a subdivision technique and interval arithmetic in a manner similar to the one used in this paper. As in this paper, the result of Hales et al. [17] is tight in the numerical sense, which means that due to the discretization error introduced by subdividing a space over $\mathbb{R}$ into finitely many pieces, parts of the proof must be carried out by other means.

Other instances of classic geometric problems that were resolved with the help of computer-assisted proofs are a tight bound for the Erdős-Szekeres problem for the existence of convex paths in planar sets of 17 points [27], a precursor by Hass and Schlafly [19] to the proof of the double bubble theorem by Hutchings et al. [20] (the shape that encloses and separates two given volumes and has the minimum possible surface area is a standard double bubble, i.e., three spherical surfaces meeting at angles of $^{2\pi}/_{3}$ on a common disk), or the proof of NP-hardness of finding a minimum-weight triangulation (MWT) of a planar point set by Mulzer and Rote [25].

Further examples in the context of packing and covering include a branch-and-bound approach for covering of polygons by not necessarily congruent disks with prescribed centers and a minimal sum of radii by Bánhelyi et al. [5].

2 A worst-case optimal algorithm

Now we describe Layer Packing, our worst-case optimal algorithm for packing squares into the unit disk $\mathcal{D}$. The basic idea is to combine refined variants of basic techniques (such as Shelf Packing) in several directions, subdividing the packing area into multiple geometric layers and components.

2.1 Outline of Layer Packing

By $s_1, \ldots, s_n$, we denote a sequence of squares and simultaneously their side lengths and assume that $s_1 \geq \cdots \geq s_n$ is a sorted sequence. Layer Packing distinguishes three cases that depend on the sizes of the first few, largest squares; see Figure 2 for illustrations.

(C 1) If $s_1 \leq 0.295$, we place a square of side length $\mathcal{X} = 1.388$ concentric into $\mathcal{D}$ and place one square of side length $\mathcal{X}_i = 0.295$, $i \in \{1, \ldots, 4\}$, to each side of $\mathcal{X}$, see Figure 2(a). The four largest squares $s_1, \ldots, s_4$ are placed in these containers $\mathcal{X}_1, \ldots, \mathcal{X}_4$. All other squares are packed into $\mathcal{X}$ using Shelf Packing.

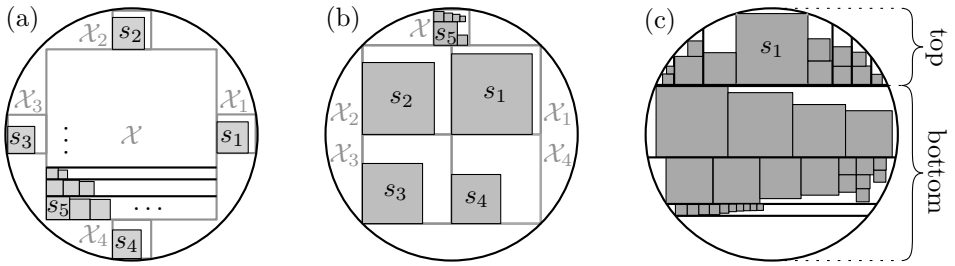


Figure 2 Illustration of the packings (a) in Case **(C1)**, (b) in Case **(C2)**, and (c) in Case **(C3)**.

(C2) If $s_1 \leq 1/\sqrt{2}$ and $s_1^2 + s_2^2 + s_3^2 + s_4^2 \geq 39/25$, let $\mathcal{X}_1, \ldots, \mathcal{X}_4$ be four squares of side length $1/\sqrt{2}$ that are placed into $\mathcal{D}$ as depicted in Figure 2(b). Furthermore, let $\mathcal{X}$ be a square of side length $\sqrt{2}/5$ that can be packed into $\mathcal{D}$ in addition to $\mathcal{X}_1, \ldots, \mathcal{X}_4$; see Figure 2(b). For $i \leq 4$, s_i is the only square packed into $\mathcal{X}_i$; this is possible because $s_i \leq s_1 \leq 1/\sqrt{2}$. All other squares are packed into $\mathcal{X}$ using Shelf Packing.

(C3) In the remaining cases, we make extensive use of a refined shelf packing approach. Specifically, the largest square s_1 is packed into $\mathcal{D}$ as high as possible, see Figures 2(c) and 3. The bottom side of s_1 induces a horizontal split of $\mathcal{D}$ into a *top* and a *bottom* part, which are then filled by two subroutines called Top Packing and Bottom Packing, described in Section 2.2. For each $i \geq 2$, we then

(C3a) use Top Packing to pack s_i if possible,

(C3b) else we use Bottom Packing to pack s_i.

In the remainder of the paper, we prove that Layer Packing only fails to pack a sequence of squares if its total area exceeds the critical bound of $8/5$.

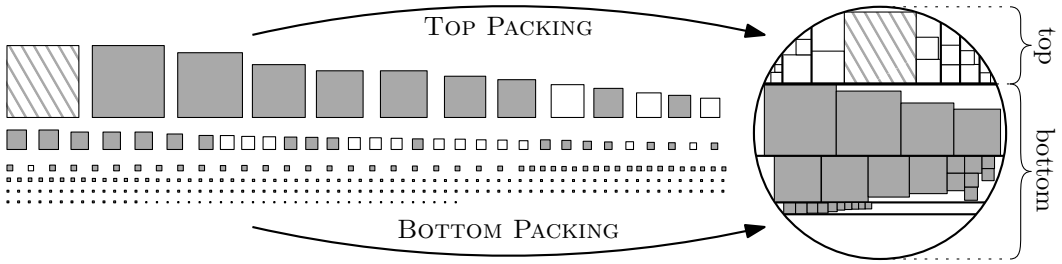


Figure 3 In case **(C3)**, the largest (hatched) square is packed topmost, inducing a top and a bottom part of $\mathcal{D}$. Subsequent (white) squares are packed into the pockets of the top part with *Top Packing* (using Refined Shelf Packing as a subroutine) if they fit; if they do not fit, they are shown in gray and packed into the bottom part with *Bottom Packing*, which uses horizontal SubContainer Slicing, and vertical Refined Shelf Packing within each slice.

2.2 Subroutines of Layer Packing

Layer Packing employs a number of different subroutines.

Refined Shelf Packing The greedy-type packing procedure Shelf Packing, employed by Moon and Moser [23], packs objects by decreasing size; see the top of Figure 1(a). At each stage, there is a (w.l.o.g. horizontal) straight cut that separates the unused portion of the container from a "shelf" into which the next square is packed. The height of a shelf is determined by the first packed object. Subsequent objects are packed next to

each other, until an object no longer fits into the current shelf; in this case, a new shelf is opened on top of the previous one. For Layer Packing, we use two modifications.
(1) Parts of the shelf boundaries may be circular arcs; however, we still have a supporting straight axis-parallel boundary and a second, orthogonal straight boundary.
(2) Our refined shelf packing uses the axis-parallel boundary line of a shelf as a support line for packing squares; in case of a collision with the circular boundary, we may move a square towards the middle of a shelf if this allows packing it. Note that this may only occur in shelves containing the horizontal diameter.

Top Packing The first and largest square s_1 is packed as high as possible into $\mathcal{D}$; see Figure 4(a). Then the horizontal line through the bottom of s_1 cuts the container into a *top part* that contains s_1, with two congruent empty pockets C_ℓ and C_r left and right of s_1; and a *bottom part*. Each pocket has two straight axis-parallel boundaries, b_x and b_y. By σ, we denote the largest square that fits into either pocket. For large s_1, the bottom side of σ does not lie on the same height as the bottom side of s_1; in that case, we ignore the parts of C_ℓ and C_r that lie below σ; see Figure 4(e). We use Refined Shelf Packing with shelves parallel to the shorter boundary among b_x and b_y, as shown in Figure 4(b) and (c). If a square does not fit into either pocket, it is packed into the bottom part.

Bottom Packing A square that does not fit into the top part of $\mathcal{D}$ is packed into the bottom part. For this purpose, we use (horizontal) SubContainer Slicing, and (vertical) SubContainer Packing within each subcontainer; see Figure 3 for the overall picture.

SubContainer Slicing For packing squares in the bottom part of $\mathcal{D}$, SubContainer Slicing subdivides $\mathcal{D}$ into smaller containers C_i, by using straight horizontal cuts; see Figure 4(a). The height of a subcontainer is determined by the first square packed into it.

SubContainer Packing Within each subcontainer, we use Refined Shelf Packing with vertical shelves. These shelves are packed from the longer of the two horizontal cuts, i.e., to pack C_i, we start from the boundary that is closer to the disk center; see Figure 4(d).

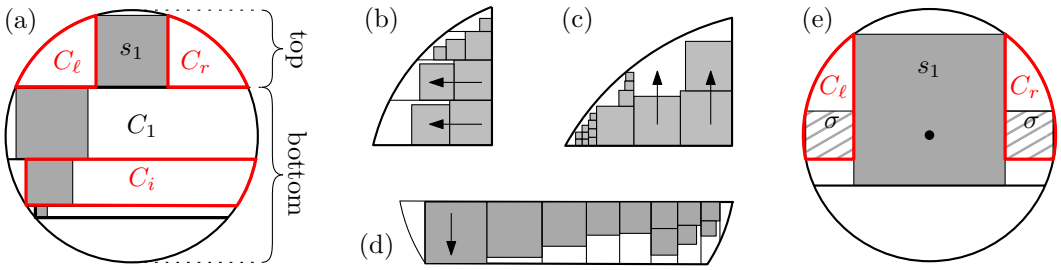


Figure 4 (a) Packing s_1 topmost into $\mathcal{D}$ yields the top part of $\mathcal{D}$ with pockets C_ℓ and C_r, and the bottom part of $\mathcal{D}$. The bottom part is partitioned by SubContainer Slicing into subcontainers C_i, with heights corresponding to the first packed square. (b) A pocket C_ℓ for which $b_x \leq b_y$ implies horizontal shelf packing. (c) A pocket C_ℓ for which $b_x > b_y$ implies vertical shelf packing. (d) Within each subcontainer C_i, SubContainer Packing places squares along vertical shelves, starting from the longer straight cut of C_i. (e) For large s_1, we disregard the parts of C_ℓ and C_r that lie below their inscribed square σ.

3 Proofs based on interval arithmetic

In interval arithmetic, operations like addition, multiplication or taking the square root are performed on real intervals $[a, b] \subset \mathbb{R}$ instead of real numbers. When applied to intervals, an operation $[a_1, b_1] \circ [a_2, b_2]$ results in the smallest interval that contains all possible values

of $x \circ y$ for $x \in [a_1, b_1], y \in [a_2, b_2]$. In a practical implementation on computers with finite precision, computing the smallest such interval is not always possible. However, using appropriate rounding modes or error bounds, it is still possible to compute an interval that over-approximates the resulting interval, i.e., contains all possible outcomes of the corresponding real operation. Predicates such as $[a_1, b_1] \leq [a_2, b_2]$ can also be evaluated on intervals. The result is a subset of $\{\texttt{false}, \texttt{true}\}$ containing all possible outcomes of $x \leq y$ for $x \in [a_1, b_1], y \in [a_2, b_2]$. This allows evaluating quantifier-free formulas on the Cartesian product of intervals in an over-approximative way.

In many cases throughout this paper, we want to prove that a given non-linear system of real constraints over a bounded k-dimensional space $\mathcal{R}$ is unsatisfiable. This space is typically spanned by a set of k real variables. Conceptually, to do this in an automatic fashion, we subdivide $\mathcal{R}$ into a sufficiently large number of k-dimensional cuboids. Each such cuboid is defined by an interval for each of the k real variables spanning $\mathcal{R}$. We then apply interval arithmetic to each such cuboid $\mathcal{C}$ to find a set S of constraints that together eliminate all points of $\mathcal{C}$, thus proving that no point in $\mathcal{C}$ satisfies all our constraints for a counterexample.

We use this simple technique because it scales relatively well with the complexity of the constraints and can handle non-polynomial constraints involving functions such as $\arccos(x)$ that occur in some of our proofs.

To improve the efficiency of this approach, our implementation of the basic concept is optimized in several ways. For instance, the subdivision proceeds in a tree-like fashion according to a fixed ordering of the k variables $v_1, \ldots, v_k$ spanning $\mathcal{R}$. If variables $v_1, \ldots, v_j$ suffice to exclude a part of $\mathcal{R}$, we do not split $v_{j+1}, \ldots, v_k$ on that part. Furthermore, we adaptively increase the local fineness of our subdivision if a coarser subdivision does not suffice for some part of $\mathcal{R}$.

Overall, this leads to a limited number of automated proofs[1]; for some of these, manual checking would also be feasible, but would involve many case distinctions and would be tedious and unsatisfying. Instead, we replace these proofs by the automatic procedure outlined above to have a clear structure instead of an otherwise overwhelming set of arguments. Combined, all automatic proofs required for this paper take less than 1.5 hours and less than 300 MB of memory on the 4 physical cores of the 2.3 GHz Intel i5-8259U CPU in one of the authors' laptops. The proofs involve up to $k = 9$ variables.

4 Analysis of subroutines

In the following, we establish a number of bounds for the subroutines from Section 2.2 that we use to prove the performance guarantee for Layer Packing.

4.1 Shelf Packing

In several places we make use of the following classic result regarding Shelf Packing.

▶ **Lemma 2** ([23]). *Shelf Packing packs every sequence $t_1 \geq \cdots \geq t_u$ of squares with a total area of at most $1/2 \cdot hw$ into an $h \times w$-rectangle with $t_1 \leq h \leq w$.*

If the side length of the largest square is small compared to the size of the container, one can guarantee a higher packing density.

[1] Source code available at `https://github.com/phillip-keldenich/squares-in-disk`.

▶ **Lemma 3** ([23]). *Any finite set of squares with largest square $x_1 < 1/2$ is packed by* Shelf Packing *into a unit square, provided its total area is at most $1/2 + 2(x_1 - 1/2)^2$.*

4.2 Top Packing

We prove the following lower bound on the square area packed by Top Packing, as long as at least one square fits into the left pocket C_ℓ. By $\sigma = \sigma(s_1)$, we denote the side length of the largest square that can be packed into C_ℓ or C_r; see Figure 5(c)–(e).

▶ **Lemma 4.** *Let $s_1 \geq \cdots \geq s_n$ be a sequence of squares for which* Layer Packing *fails to pack s_n. If $s_n \leq \sigma$, then* Top Packing *packs squares of total area at least $0.83\sigma^2$.*

Intuitively, the proof makes use of the Shelf Packing bound on the squares inscribed in C_ℓ and C_r, but additionally uses the gaps in C_r and C_ℓ to bound the square area packed into each of C_ℓ and C_r by $0.415\sigma^2$.

Before presenting its proof, we make some helpful observations. We assume the center of our unit disk $\mathcal{D}$ lies at the origin $(0, 0)$ of our coordinate system. Recall that Top Packing packs the largest square s_1 as high as possible into $\mathcal{D}$. This implies that the center of s_1 is on the vertical line $x = 0$. For some $u \in (-1, 1)$, we denote by $T(u)$ the side length of the largest square with center on $x = 0$ and bottom on $y = u$ that fits into $\mathcal{D}$; see Figure 5(a).

The inverse function $T^{-1}(s)$ of $T(u)$ describes the highest possible y-coordinate of the bottom side of a square of side length s; see Figure 5(b). Thus, Top Packing places the bottom-left corner of s_1 at $(-s_1/2, T^{-1}(s_1))$; note that this can be below or above the center of $\mathcal{D}$. Furthermore, recall that Top Packing packs the remaining disks into the pockets C_ℓ and C_r induced by placing s_1; see Figure 5(c).

Now, we present explicit formulas. Solving the equations in Figure 5(a)–(b), we get

$$T(u) = 2/5 \cdot \left(\sqrt{5 - u^2} - 2u\right) \quad \text{and} \quad T^{-1}(s) = \sqrt{1 - 1/4 \cdot s^2} - s.$$

To compute $\sigma(s_1)$, we observe the following. Below some threshold s_1^*, the bottom side of the inscribed square of C_ℓ lies on the horizontal line $y = T^{-1}(s_1)$ and its top left corner touches $\mathcal{D}$; see Figure 5(c). At the threshold $s_1^* = \sqrt{1/3 \cdot (2 + \sqrt{2})}$, both left corners of

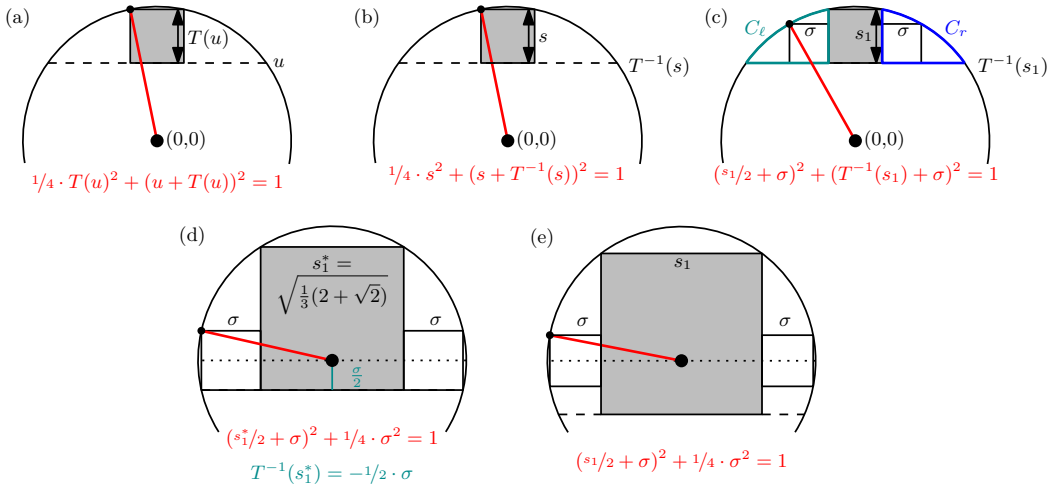


Figure 5 (a) The definition of $T(u)$ and its defining equation. (b) The definition of $T^{-1}(s)$ and its equation. (c) The pockets C_ℓ and C_r used by Top Packing with their inscribed square σ and its equation if $s_1 < s_1^*$, (d) $s_1 = s_1^*$ and (e) $s_1 > s_1^*$.

the inscribed square of C_ℓ touch the disk; see Figure 5(d). For $s_1 > s_1^*$, the center of C_ℓ's inscribed square lies on $y = 0$ and both left corners touch the disk; see Figure 5(e). Overall, this yields

$$\sigma = \begin{cases} 1/4 \cdot \left(-s_1 - 2T^{-1}(s_1) + \sqrt{8 - (s_1 - 2T^{-1}(s_1))^2}\right), & \text{if } s_1 \leq s_1^*, \\ 1/5 \cdot \left(\sqrt{20 - s_1^2} - 2s_1\right), & \text{otherwise.} \end{cases}$$

Now we are ready to present a proof of Lemma 4.

Proof of Lemma 4. We begin by observing that $s_n \leq \sigma$ would fit into either C_ℓ or C_r; as we fail to pack s_n, C_ℓ and C_r must contain other squares. In the following, we prove that Top Packing packs squares of area $A \geq 0.415\sigma^2$ into C_ℓ. An analogous argument works for C_r, implying an overall bound of $0.83\sigma^2$.

By $\ell_1 := \sqrt{1 - T^{-1}(s_1)^2} - s_1/2$, we denote the length of the bottom boundary of C_ℓ; see Figure 6(a). W.l.o.g., we assume $\ell_1 \leq s_1$; the other case is symmetric. In other words, we assume that the bottom boundary of C_ℓ is shorter than its right boundary, which means that we are using horizontal shelves that we fill from right to left as depicted in Figure 4(b).

Consider the subsequence $t_1, \ldots, t_u, t_{u+1}$ of $s_1, \ldots, s_n$, where $t_1, \ldots, t_u$ are the squares packed by Top Packing into C_ℓ before height σ is (strictly) exceeded, and t_{u+1} is the next square that we try to pack into C_ℓ. We observe that t_{u+1} may or may not be packed into C_ℓ by Top Packing, and that $u \geq 1$ by $t_1 \leq \sigma$, i.e., after placing the first square, height σ is not exceeded. We make use of the following lemma, proved by interval arithmetic.

▶ **Lemma 5** (Automatic Analysis for Top Packing). *Let $\ell_1 \leq s_1$, $x_{+1} = s_1/2 + \sigma/2\sqrt{2}$ and $x_{+2} = s_1/2 + 0.645\sigma$. Furthermore, let*

$$y_{+1} = \begin{cases} T^{-1}(s_1) + \sigma + \sigma/2\sqrt{2}, & \text{if } s_1 \leq s_1^*, \\ \sigma/2 + \sigma/2\sqrt{2}, & \text{otherwise,} \end{cases} \quad y_{+2} = \begin{cases} T^{-1}(s_1) + 2 \cdot 0.645\sigma, & \text{if } s_1 \leq s_1^*, \\ -\sigma/2 + 2 \cdot 0.645\sigma, & \text{otherwise;} \end{cases}$$

see Figure 6. Let $F_{TP_1}(s_1) := x_{+1}^2 + y_{+1}^2$ and $F_{TP_2}(s_1) := x_{+2}^2 + y_{+2}^2$. Then, for all $0.295 \leq s_1 \leq \sqrt{8/5}$, we have (1) $F_{TP_1}(s_1) \leq 1$ and (2) $F_{TP_2}(s_1) \leq 1$.

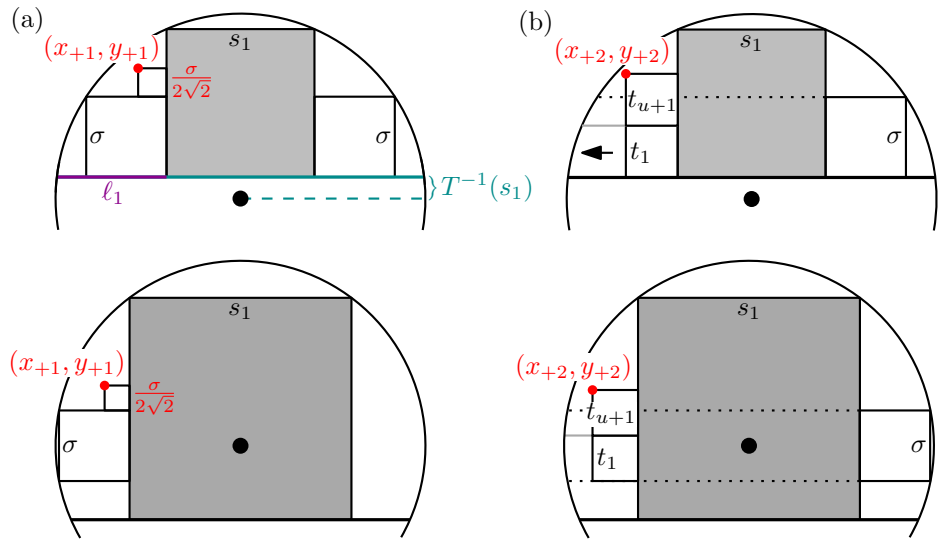


Figure 6 Illustration of the proof of Lemma 5. In both parts, the red point is always contained in the disk $\mathcal{D}$. (a) The values occurring in part (1). (b) The situation for t_1, t_{u+1} of maximum possible size in part (2).

If $0.645\sigma \le t_1$, the packed area inside C_ℓ is at least $t_1^2 \ge 0.645^2\sigma^2 > 0.415\sigma^2$. Thus, in the following, we assume $t_1 < 0.645\sigma$.

Furthermore, if $t_{u+1} \le \sigma/2\sqrt{2}$, we can apply Lemma 5 (1), showing that t_{u+1} can be packed into C_ℓ by Top Packing; see Figure 6(a). In particular, t_{u+1} can always be packed into C_ℓ such that its bottom side lies on height σ and its right side touches s_1. The total area packed by Top Packing into C_ℓ is at least the total area packed by Shelf Packing into the square of area σ; here we use the fact that the height of the bottom segment of a pocket and the bottom segment of the contained square σ coincide, see also Figure 4(e). Because packing t_{u+1} exceeds height σ, Lemma 3 implies that the total area of t_{u+1} and the squares already packed into C_ℓ exceeds $\sigma^2/2$. Thus, in the following, we assume $\sigma/2\sqrt{2} < t_{u+1} \le t_1 < 0.645\sigma$.

If $t_1 \le \sigma/2$, at least four squares are packed into C_ℓ by Refined Shelf Packing before height σ is exceeded. Consequently, the total packed area is at least $4\,(\sigma/2\sqrt{2})^2 = \sigma^2/2$. Thus, in the following we assume $t_1 > \sigma/2$.

Now let us assume that only one shelf is constructed before height σ is exceeded. That shelf has height t_1 and thus we must have $t_{u+1} > \sigma - t_1$. We use Lemma 5 (2) to prove that we can pack t_{u+1} on top of the first shelf, assuming that $t_1 = t_{u+1} = 0.645\sigma$ are as large as possible; see Figure 6(b). Thus, the total area packed into C_ℓ is at least $t_1^2 + t_{u+1}^2 \ge t_1^2 + (\sigma - t_1)^2 \ge \sigma^2/2$.

Otherwise, at least two shelves are constructed before height σ is exceeded. The first shelf has height t_1. The second shelf contains at least two squares because its height is at most $\sigma - t_1 \le \sigma/2$, and thus at most half of its width. Thus, the area packed into C_ℓ is at least $t_1^2 + 2t_{u+1}^2 \ge \sigma^2/4 + 2(\sigma/2\sqrt{2})^2 = \sigma^2/2$, concluding the proof of Lemma 4. ◀

4.3 Subcontainer Packing

For the analysis of SubContainer Packing, let $C_1, \dots, C_k$ be the subcontainers constructed by Bottom Packing and let $R_1, \dots, R_k$ be the maximal rectangles contained in $C_1, \dots, C_k$; see Figure 7.

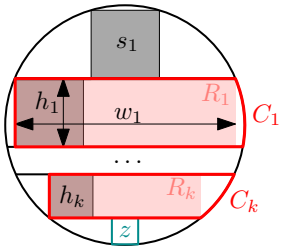


Figure 7 Subcontainers C_i, $i \le k$, produced by SubContainer Slicing.

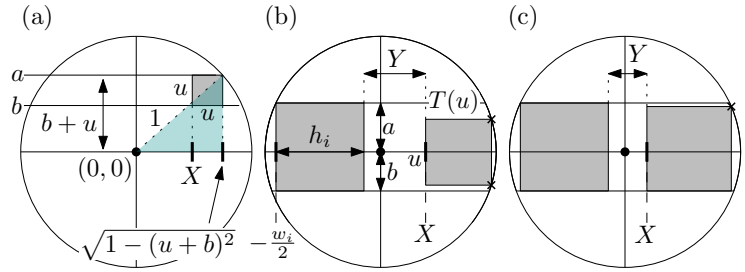


Figure 8 The computation of X for a square of side length u: (a) in case $b \ge 0$, which is symmetric to $a < 0$; (b) in case $a > 0 > b$ and $u \le 2c$; (c) in case $a > 0 > b$ and $u > 2c$.

For $i = 1, \dots, k$, let h_i and w_i denote the height and the width of R_i. Recall that h_i simultaneously denotes the height of C_i and the first square packed into C_i. Let z be the largest square that could be packed below C_k. We define $h_{k+1} := s_n$, so h_{i+1} always denotes the first square that did not fit into C_i. Furthermore, we denote the total area of squares packed into C_i by $\|C_i\|$. We establish several lower bounds on this area $\|C_i\|$. One such bound is derived from the following observation.

▶ **Observation 6.** *The total area packed by SubContainer Packing into C_i is at least the total area packed by Shelf Packing into R_i.*

If the width of R_i is at least twice its height, the following lemma improves on this bound. A proof can be found in the full version of this paper.

▶ **Lemma 7.** *For every sequence of squares $s_1, \dots, s_n$ for which* Layer Packing *constructs at least i subcontainers and fails to pack s_n, if $w_i \geq 2h_i$, we can bound the area packed into C_i by*

$$\|C_i\| \geq B_1(h_i, w_i, h_{i+1}) := \max \begin{cases} 1/2 \cdot h_i w_i + 1/4 \cdot h_i^2, \\ h_i^2 + (w_i - h_i - h_{i+1})h_{i+1}, \\ 1/2 \cdot h_i(w_i + h_i) - h_{i+1}^2. \end{cases}$$

We always pack at least the square h_i into C_i. As $h_{i+1} \leq h_i$, if $h_i + h_{i+1} \leq w_i$, we pack at least two squares into C_i: Let s_j be the second square we consider packing into C_i. If h_i and s_j do not fit into C_i, then $s_j = h_{i+1}$, as we would open a new subcontainer for s_j. This contradicts $h_i + h_{i+1} \leq w_i$, as h_i and h_{i+1} fit into R_i and thus into C_i. Summarizing, we can extend B_1 as follows.

▶ **Lemma 8.** *For every sequence of squares $s_1, \dots, s_n$ for which* Layer Packing *constructs at least i subcontainers and fails to pack s_n, we can bound the area packed into C_i by*

$$\|C_i\| \geq B_2(h_i, w_i, h_{i+1}) := \begin{cases} h_i^2 & \text{if } w_i < h_i + h_{i+1}, \\ h_i^2 + h_{i+1}^2 & \text{if } h_i + h_{i+1} \leq w_i < 2h_i, \\ B_1(h_i, w_i, h_{i+1}) & \text{if } 2h_i \leq w_i. \end{cases}$$

For the last lemma of this subsection, we introduce some useful notation. Let a, b be the y-coordinates of the upper and the lower side of C_i and let $c := c(a, b) = \min\{a, -b\}$. When C_i contains the center of the disk, i.e., $a > 0 > b$, c denotes the distance of the origin to the nearer side of C_i, see Figure 8(b) and (c). The maximal x-coordinate X of the left side of a square of side length u in C_i is determined by

$$X(a, b, u) := \left\{ \begin{array}{ll} \sqrt{1 - (u + b)^2} - u & \text{if } b \geq 0, \\ \sqrt{1 - (u - a)^2} - u & \text{if } a < 0, \\ T^{-1}(u) & \text{else if } u \leq 2c, \\ \sqrt{1 - (u - c)^2} - u & \text{otherwise} \end{array} \right\} = \begin{cases} T^{-1}(u) & \text{if } u \leq 2c, \\ \sqrt{1 - (u - c)^2} - u & \text{otherwise} \end{cases}$$

The x-coordinate of the right side of the first square h_i packed into subcontainer C_i is $-1/2 \cdot w_i + h_i$. Thus, as h_{i+1} did not fit into C_i, we can lower bound the total width of squares packed into C_i after h_i, see Figure 9(a), by

$$Y := Y(a, h_i, w_i, h_{i+1}) := 1/2 \cdot w_i - h_i + X(a, a - h_i, h_{i+1}).$$

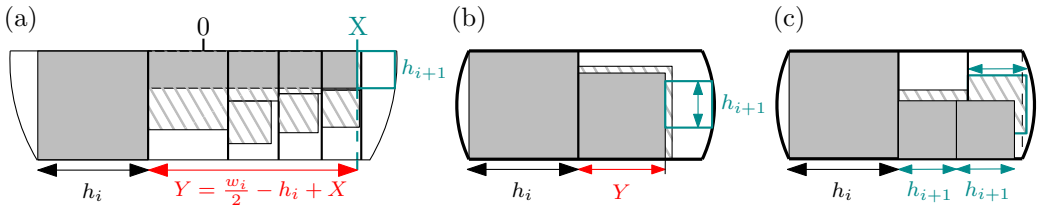


Figure 9 (a) Definition of Y. (b) The lower bound $B_3(a, h_i, w_i, h_{i+1})$ when two squares are packed into C_i. (c) The lower bound $B_3(a, h_i, w_i, h_{i+1})$ when at least three squares packed into C_i.

▶ **Lemma 9.** *For every sequence of squares $s_1, \dots, s_n$ for which* Layer Packing *constructs at least i subcontainers and fails to pack s_n, we can bound the area packed into C_i by*

$$\|C_i\| \geq B_3(a, h_i, w_i, h_{i+1}) := \max \begin{cases} h_i^2 + \max\{0, Y(a, h_i, w_i, h_{i+1})\} \cdot h_{i+1}, & (8.1) \\ h_i^2 + \min\left(\max^2(Y(a, h_i, w_i, h_{i+1}), 0), 2h_{i+1}^2\right). & (8.2) \end{cases}$$

Proof. If $Y \leq 0$, both bounds (8.1) and (8.2) simplify to h_i^2 and are valid, because h_i is packed into C_i. Thus, let us assume $Y > 0$. This implies that there are at least two squares packed into C_i; otherwise, h_{i+1} would fit into C_i. As Y is a lower bound on the total width of squares packed into C_i after h_i and h_{i+1} is a bound on their height, we obtain bound (8.1); see Figure 9(a).

Furthermore, if exactly two squares are packed into C_i, Y^2 can be used as lower bound on the area of the second square, see Figure 9(b). Otherwise, at least three squares are packed into C_i, and we can use $2h_{i+1}^2$ to bound their area, see Figure 9(c). Combining these two cases yields bound (8.2). ◀

We combine these previous bounds into a general lower bound for $\|C_i\|$.

▶ **Corollary 10.** *For every sequence of squares $s_1, \dots, s_n$ for which* Layer Packing *constructs at least i subcontainers and fails to pack s_n, we can bound the area packed into C_i by*

$$\|C_i\| \geq B_4(a, h_i, w_i, h_{i+1}) := \max \begin{cases} B_2(h_i, w_i, h_{i+1}), & \textit{(Lemma 8)} \\ B_3(a, h_i, w_i, h_{i+1}). & \textit{(Lemma 9)} \end{cases}$$

5 Analysis of the main algorithm

In this section, we prove our main result using the tools provided in Sections 3 and 4. On the highest level, the proof consists of three parts corresponding to the three cases that our algorithm distinguishes.

5.1 Analysis of (C1)

Recall that in case **(C 1)**, we place a container square $\mathcal{X}$ of side length 1.388 into $\mathcal{D}$, and pack the first four squares into pockets outside $\mathcal{X}$ and all remaining disks into $\mathcal{X}$ using Shelf Packing; see Figure 2(a).

▶ **Lemma 11.** *If* Layer Packing *fails to pack a sequence of squares $s_1, \dots, s_n$ with $s_1 \leq 0.295$, the total area of the squares exceeds $8/5$.*

Proof. Consider scaling down all side lengths by a factor of $1/1.388$, such that $\mathcal{X}$ is the unit square and $s_1 \leq 0.295/1.388 \approx 0.2125$. As s_5 is the first square packed by Shelf Packing into $\mathcal{X}$, Lemma 3 implies that the total area packed into the scaled $\mathcal{D}$ is at least $f(s_5) = 4s_5^2 + 1/2 + 2(s_5 - 1/2)^2$ with derivative $f'(s_5) = 12s_5 - 2$, which is minimized for $s_5 = 1/6$, where $f(1/6) = 5/6$. Thus, in the non-scaled configuration, the area packed is at least $5/6 \cdot 1.388^2 = 120409/75000 \approx 1.605 > 8/5$, concluding the proof. ◀

5.2 Analysis of (C2)

Recall that in case **(C 2)**, we pack the four largest squares into the squares $\mathcal{X}_1, \dots, \mathcal{X}_4$; all other squares are packed into a square container $\mathcal{X}$ on top of them; see Figure 2(b).

▶ **Lemma 12.** *If* Layer Packing *fails to pack a sequence $s_1, \dots, s_n$ of squares with $0.295 < s_1 \leq 1/\sqrt{2}$ and $s_1^2 + s_2^2 + s_3^2 + s_4^2 \geq 39/25$, the total area of the squares exceeds $8/5$.*

Proof. By assumption, the total area of the squares s_1, s_2, s_3, s_4 is at least $39/25 = 8/5 - 1/25$. As $\mathcal{X}$ has an area of $2/25$, Lemma 2 implies that Shelf Packing (and thus Layer Packing) only fails to pack all remaining squares into $\mathcal{X}$ if their area exceeds $1/25$. Consequently, the total area of the squares exceeds $8/5$, concluding the proof. ◀

5.3 Analysis of (C3)

Recall that z denotes the largest square that could be packed below the last subcontainer C_k constructed by Bottom Packing, as illustrated in Figure 7 or in Figure 10, where we have reflected the instance along the x-axis. We consider a sequence $s_1, \ldots, s_n$ of squares of total area S that Layer Packing fails to pack, and assume w.l.o.g. that s_{n-1} is packed. This implies $z < s_n$, where z denotes the side length of the largest square that could be packed below the last subcontainer C_k constructed by Bottom Packing; otherwise, a further subcontainer is constructed. We have $z = T(-T^{-1}(s_1) + \sum_{i=1}^{k} h_i)$; see Figure 10(a).

By $w(y_t, h) = 2\sqrt{\min\{1 - y_t^2, 1 - (y_t - h)^2\}}$, we denote the maximum width of a rectangle R that can be placed in $\mathcal{D}$ with top side at $y = y_t$ and height h; see Figure 10(b). Thus, we can express the width w_i of the rectangle R_i inscribed in some subcontainer C_i in terms of $s_1, h_1, \ldots, h_i$ as

$$w_i \coloneqq w\left(\left(T^{-1}(s_1) - \sum_{j=1}^{i-1} h_j\right), h_i\right).$$

Recall that $\sigma \coloneqq \sigma(s_1)$ denotes the side length of the largest squares that fits into the pockets C_ℓ and C_r as illustrated in Figure 5(c). In order to distinguish whether Top Packing has packed any squares into the pockets, we consider the function

$$E(s_1, s_n) \coloneqq \begin{cases} 0.83 \cdot \sigma(s_1)^2, & \text{if } s_n \leq \sigma, \\ 0, & \text{otherwise,} \end{cases}$$

which describes the total square area that Top Packing is guaranteed to pack due to Lemma 4.

For the analysis of Case **(C3)**, we distinguish cases depending on the number k of subcontainers constructed by Bottom Packing. Specifically, we consider the cases $k = 0, k = 1, k \in \{2, 3, 4\}$, and $k \geq 5$.

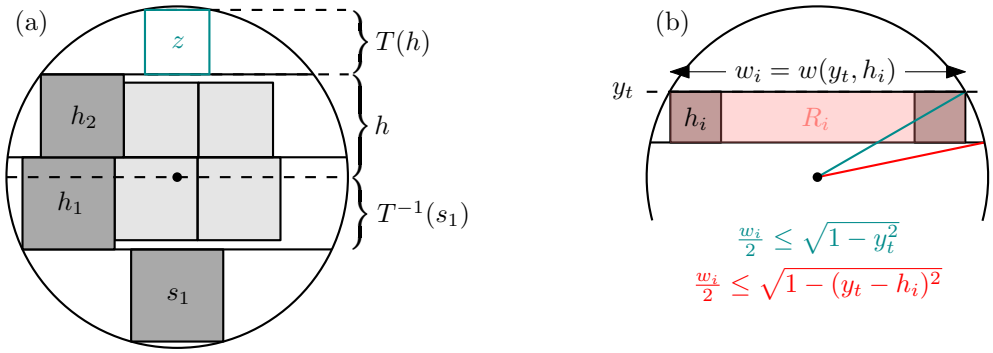


Figure 10 (a) Computing z for $k = 2$ using functions T and T^{-1}. (b) Computing the width $w_i = w(y_t, h)$ of rectangle R_i.

5.3.1 Analysis for no subcontainer

▶ **Lemma 13.** *If* Layer Packing *fails to pack a sequence $s_1, \dots, s_n$ of squares and* Bottom Packing *does not construct a subcontainer, the total area of the squares exceeds $8/5$.*

Proof. Because the algorithm fails to construct a first subcontainer in the bottom part, it follows that placing s_n as far to the bottom as possible yields an overlap with s_1. However, the minimum value for $s_1^2 + s_n^2$ for two overlapping squares packed into a disk is attained for $s_1 = s_n$. This corresponds to the worst-case configuration, implying that the total area of s_1 and s_n exceeds $8/5$. ◀

5.3.2 Analysis for one subcontainer

▶ **Lemma 14.** *If* Layer Packing *fails to pack a sequence $s_1, \dots, s_n$ of squares and* Bottom Packing *constructs exactly one subcontainer, the total area of the squares exceeds $8/5$.*

Proof. Combining Lemma 4 and Corollary 10 allows us to bound the area of $s_1, \dots, s_n$ by

$$S \geq F_{SC_1}(s_1, h_1, s_n) \coloneqq s_1^2 + B_4(T^{-1}(s_1), h_1, w_1, s_n) + s_n^2 + E(s_1, s_n),$$

where $E(s_1, s_n) = 0.83 \cdot \sigma(s_1)^2$ if $s_n \leq \sigma$, and $E(s_1, s_n) = 0$ if $s_n > \sigma$. Furthermore, we know $0 < z < s_n$, because Layer Packing fails to pack s_n. Moreover, we claim that at least one of the following conditions must hold: $s_1 > 1/\sqrt{2}$, $w_1 < 2h_1$, or $s_1^2 + h_1^2 + 2s_n^2 < 39/25$. Assume for contradiction that neither of these conditions hold. By $w_1 \geq 2h_1$, we know that at least two squares are packed into the first subcontainer. One of these squares has area h_1^2, and the other has area at least s_n^2. In particular, this implies that the algorithm packs s_1, s_2 and s_3. This implies $s_2 \geq h_1$ and $s_3, s_4 \geq s_n$. Thus we have $s_1^2 + s_2^2 + s_3^2 + s_4^2 \geq s_1^2 + h_1^2 + 2s_n^2 \geq 39/25$, which together with $s_1 \leq 1/\sqrt{2}$ implies that we are in Case **(C 2)** of our algorithm. This is a contradiction, because we only construct subcontainers in Case **(C 3)**. Thus the following lemma, proved automatically using interval arithmetic, proves that these conditions are sufficient to ensure $S \geq 8/5$.

▶ **Lemma 15** (One Subcontainer, Automatic Analysis for Lemma 14)**.** *Let $z \coloneqq T(T^{-1}(s_1) + h_1)$. For all s_1, h_1, s_n with $0 < z < s_n \leq h_1 \leq s_1$, $h_1 \leq T^{-1}(s_1) + 1$ and*

$$(s_1 > 1/\sqrt{2}) \vee (w_1 < 2h_1) \vee \left(s_1^2 + h_1^2 + 2s_n^2 < 39/25\right),$$

we have $F_{SC_1}(s_1, h_1, s_n) > 8/5$. ◀

5.3.3 Analysis for two to four subcontainers

▶ **Lemma 16.** *If* Layer Packing *fails to pack a sequence $s_1, \dots, s_n$ of squares and* Bottom Packing *constructs $k \in \{2, 3, 4\}$ subcontainers, the total area of the squares exceeds $8/5$.*

Proof. We use similar ideas as in the proof of Lemma 14. We bound the area packed by Top Packing by $E(s_1, s_n)$ using Lemma 4. Furthermore, we use Corollary 10 to bound the area packed into each of the $k \in \{2, 3, 4\}$ subcontainers by

$$\|C_i\| \geq B_4\left(\left(T^{-1}(s_1) - \sum_{j=1}^{i-1} h_i\right), h_i, w_i, h_{i+1}\right), 1 \leq i \leq k,$$

where $h_{k+1} \coloneqq s_n$. We can express $w_i = w(T^{-1}(s_1) - \sum_{j=1}^{i-1} h_j, h_i)$ in terms of s_1 and $h_j, 1 \leq j \leq i$. In total, for k subcontainers, this yields the bound

$$S \geq F_{SC_k}(s_1, h_1, \dots, h_k, s_n) \coloneqq s_1^2 + s_n^2 + E(s_1, s_n) + \sum_{i=1}^{k} B_4\left(T^{-1}(s_1) - \sum_{j=1}^{i-1} h_i, h_i, w_i, h_{i+1}\right).$$

Finally, we know $0 < z < s_n$ because the algorithm fails to pack s_n. Thus, the following lemma, proved automatically using interval arithmetic, suffices to complete the proof of Lemma 16.

▶ **Lemma 17** (Automatic Analysis for Lemma 16). *Let* $z_k = T(-T^{-1}(s_1) + \sum_{i=1}^{k} h_i)$.

($k = 2$) *For all* s_1, h_1, h_2, s_n *with* $0 < z_2 < s_n \leq h_2 \leq h_1 \leq s_1$ *and* $0.295 \leq s_1 \leq \sqrt{8/5}$ *and* $h_1 + h_2 \leq 1 + T^{-1}(s_1)$, *we have* $F_{SC_2} > 8/5$.

($k = 3$) *For all* s_1, h_1, h_2, h_3, s_n *with* $0 < z_3 < s_n \leq h_3 \leq h_2 \leq h_1 \leq s_1$ *and* $0.295 \leq s_1 \leq \sqrt{8/5}$ *and* $h_1 + h_2 + h_3 \leq 1 + T^{-1}(s_1)$, *we have* $F_{SC_3} > 8/5$.

($k = 4$) *For all* $s_1, h_1, h_2, h_3, h_4, s_n$ *with* $0 < z_4 < s_n \leq h_4 \leq h_3 \leq h_2 \leq h_1 \leq s_1$ *and* $0.295 \leq s_1 \leq \sqrt{8/5}$ *and* $h_1 + h_2 + h_3 + h_4 \leq 1 + T^{-1}(s_1)$, *we have* $F_{SC_4} > 8/5$. ◀

Due to space constraints, we omit the detailed analysis for five or more subcontainers, which can be found in the full version of the paper. This completes the analysis of **(C 3)** and thus the proof of our main result.

6 Conclusion

We have established the critical density for packing squares into a disk: Any set of squares of total area at most $8/5$ can be packed into a unit disk. As shown by our lower bound example, this guarantee is best-possible, i.e., it cannot be improved. The proof is based on an algorithm that subdivides the disk into horizontal subcontainers and uses a refined shelf packing scheme. The correctness of this algorithm is shown by careful manual analysis, complemented by a computer-assisted part that is based on interval arithmetic.

There is a variety of interesting directions for future research. Of particular interest is the critical density for packing squares of bounded size into a disk, which will result in a higher packing density; a more general problem concerns the critical packing density for packing other types of objects of bounded size into other types of containers. Other questions arise from considering questions in three- or even higher-dimensional space. We are optimistic that many of our techniques will be useful for settling these problems.

References

1. Mikkel Abrahamsen, Tillmann Miltzow, and Nadja Seiferth. Framework for ∃ℝ-completeness of two-dimensional packing problems. *IEEE Symposium on Foundations of Computer Science (FOCS)*, 2020. accepted. `arXiv:2004.07558`.
2. K. Appel and W. Haken. Every planar map is four colorable. Part I. Discharging. *Illinois Journal of Mathematics*, 21:429–490, 1977.
3. K. Appel and W. Haken. Every planar map is four colorable. Part II. Reducibility. *Illinois Journal of Mathematics*, 21:491–567, 1977.
4. Archimedes. Measurement of a circle, 250 B.C.
5. Balázs Bánhelyi, Endre Palatinus, and Balázs L. Lévai. Optimal circle covering problems and their applications. *Central European Journal of Operations Research*, 23:815–832, 2015.
6. Aaron T. Becker, Sándor P. Fekete, Phillip Keldenich, Sebastian Morr, and Christian Scheffer. Packing Geometric Objects with Optimal Worst-Case Density. In *Symposium on Computational Geometry (SoCG)*, pages 63:1–63:6, 2019. Video available at `https://www.ibr.cs.tu-bs.de/users/fekete/Videos/PackingCirclesInSquares.mp4`. `doi:10.4230/LIPIcs.SoCG.2019.63`.
7. Károly Böröczky Jr. *Finite packing and covering*, volume 154. Cambridge University Press, 2004.

8 Peter Brass, William O.J. Moser, and János Pach. *Research problems in discrete geometry*. Springer Science & Business Media, 2006.

9 E. D. Demaine, S.P. Fekete, and R. J. Lang. Circle packing for origami design is hard. In *Origami^5: 5th International Conference on Origami in Science, Mathematics and Education*, AK Peters/CRC Press, pages 609–626, 2011. arXiv:1105.0791.

10 Gábor Fejes Tóth. Recent progress on packing and covering. *Contemporary Mathematics*, 223:145–162, 1999.

11 S. P. Fekete and J. Schepers. New classes of fast lower bounds for bin packing problems. *Mathematical Programming*, 91(1):11–31, 2001.

12 S. P. Fekete and J. Schepers. A general framework for bounds for higher-dimensional orthogonal packing problems. *Mathematical Methods of Operations Research*, 60:311–329, 2004.

13 Sándor P. Fekete, Utkarsh Gupta, Phillip Keldenich, Christian Scheffer, and Sahil Shah. Worst-case optimal covering of rectangles by disks. In *Symposium on Computational Geometry (SoCG)*, pages 42:1–42:23, 2020.

14 Sándor P. Fekete, Phillip Keldenich, and Christian Scheffer. Packing Disks into Disks with Optimal Worst-Case Density. In *Symposium on Computational Geometry (SoCG)*, pages 35:1–35:19, 2019. doi:10.4230/LIPIcs.SoCG.2019.35.

15 Sándor P. Fekete, Sebastian Morr, and Christian Scheffer. Split packing: Algorithms for packing circles with optimal worst-case density. *Discrete & Computational Geometry*, 61(3):562–594, 2019.

16 Michael R. Garey and David S. Johnson. Complexity results for multiprocessor scheduling under resource constraints. *SIAM Journal on Computing*, 4(4):397–411, 1975.

17 Thomas Hales, Mark Adams, Gertrud Bauer, Tat Dat Dang, John Harrison, Le Truong Hoang, Cezary Kaliszyk, Victor Magron, Sean McLaughlin, Tat Thang Nguyen, Quang Truong Nguyen, Tobias Nipkow, Steven Obua, Joseph Pleso, Jason Rute, Alexey Solovyev, Thi Hoai An Ta, Nam Trung Tran, Thi Diep Trieu, Josef Urban, Ky Vu, and Roland Zumkeller. A formal proof of the Kepler conjecture. *Forum of Mathematics, Pi*, 5:e2, 2017. doi:10.1017/fmp.2017.1.

18 Thomas C. Hales. A proof of the Kepler conjecture. *Annals of Mathematics*, 162(3):1065–1185, 2005.

19 Joel Hass and Roger Schlafly. Double bubbles minimize. *Annals of Mathematics*, 151(2):459–515, 2000.

20 Michael Hutchings, Frank Morgan, Manuel Ritoré, and Antonio Ros. Proof of the double bubble conjecture. *Annals of Mathematics*, 155(2):459–489, 2002.

21 J. Y. T. Leung, T. W. Tam, C. S. Wong, G. H. Young, and F. Y. L. Chin. Packing squares into a square. *Journal of Parallel and Distributed Computing*, 10(3):271–275, 1990.

22 A. Lodi, S. Martello, and M. Monaci. Two-dimensional packing problems: A survey. *European Journal of Operational Research*, 141(2):241–252, 2002.

23 J. W. Moon and L. Moser. Some packing and covering theorems. In *Colloquium Mathematicae*, volume 17, pages 103–110. Institute of Mathematics, Polish Academy of Sciences, 1967.

24 S. Morr. Split packing: An algorithm for packing circles with optimal worst-case density. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 99–109, 2017.

25 Wolfgang Mulzer and Günter Rote. Minimum-weight triangulation is NP-hard. *Journal of the ACM*, 55(2):11:1–11:29, 2008.

26 N. Robertson, D. Sanders, P. Seymour, and R. Thomas. The four-colour theorem. *Journal of Combinatorial Theory Series B*, 70:2–44, 1997.

27 George Szekeres and Lindsay Peters. Computer solution to the 17-point Erdős-Szekeres problem. *The ANZIAM Journal*, 48(2):151–164, 2006.

28 Gábor Fejes Tóth. Packing and covering. In *Handbook of Discrete and Computational Geometry, Third Edition*, pages 27–66. Chapman and Hall/CRC, 2017.

29 R. Wilson. *Four colours suffice: How the map problem was solved*. Princeton University Press, 2013.

Sunflowers in Set Systems of Bounded Dimension

Jacob Fox
Department of Mathematics, Stanford University, CA, USA

János Pach
Alfréd Rényi Institute of Mathematics, Budapest, Hungary
Moscow Institute of Physics and Technology, Moscow, Russia

Andrew Suk
Department of Mathematics, University of California San Diego, La Jolla, CA, USA

Abstract

Given a family $\mathcal{F}$ of k-element sets, $S_1, \ldots, S_r \in \mathcal{F}$ form an *r-sunflower* if $S_i \cap S_j = S_{i'} \cap S_{j'}$ for all $i \neq j$ and $i' \neq j'$. According to a famous conjecture of Erdős and Rado (1960), there is a constant $c = c(r)$ such that if $|\mathcal{F}| \geq c^k$, then $\mathcal{F}$ contains an r-sunflower.

We come close to proving this conjecture for families of bounded *Vapnik-Chervonenkis dimension*, $\text{VC-dim}(\mathcal{F}) \leq d$. In this case, we show that r-sunflowers exist under the slightly stronger assumption $|\mathcal{F}| \geq 2^{10k(dr)^{2\log^* k}}$. Here, $\log^*$ denotes the iterated logarithm function.

We also verify the Erdős-Rado conjecture for families $\mathcal{F}$ of bounded *Littlestone dimension* and for some geometrically defined set systems.

2012 ACM Subject Classification Mathematics of computing → Combinatoric problems

Keywords and phrases Sunflower, VC-dimension, Littlestone dimension, pseudodisks

Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.37

Funding *Jacob Fox*: Supported by a Packard Fellowship and by NSF award DMS-1855635.
János Pach: Rényi Institute, Budapest and MIPT, Moscow. Supported by NKFIH grants K-176529, KKP-133864, Austrian Science Fund Z 342-N31, Ministry of Education and Science of the Russian Federation MegaGrant No. 075-15-2019-1926, ERC Advanced Grant "GeoScape."
Andrew Suk: Supported by NSF CAREER award DMS-1800746, NSF award DMS-1952786, and an Alfred Sloan Fellowship.

Acknowledgements We would like to thank Amir Yehudayoff for suggesting working with the Littlestone dimension, and the SoCG 2021 referees for helpful comments.

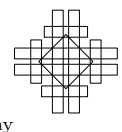
1 Introduction

An *r-sunflower* is a collection of r sets whose pairwise intersections are the same. That is, r distinct sets $S_1, \ldots, S_r$ form an r-sunflower if $S_i \cap S_j = S_{i'} \cap S_{j'}$ for all $i \neq j$ and $i' \neq j'$. For brevity, a k-element set is called a *k-set*.

Let $f_r(k)$ be the minimum positive integer m such that every family of k-sets whose size is at least m contains r members that form an r-sunflower. Erdős and Rado [11] proved that $f_r(k) \leq k!(r-1)^k$. The Erdős-Rado "sunflower conjecture" states that there is a constant $C = C(r)$ depending only on r such that $f_r(k) \leq C^k$. Over the years, some small improvements have been made on the upper bound $k!(r-1)^k$, see [1, 14]. Very recently, a breakthrough has been achieved by Alweiss, Lovett, Wu, and Zhang [5], who proved that

$$f_r(k) \leq (cr^3 \log k \log\log k)^k,$$

where c is an absolute constant. For an alternative proof of this result, using Shannon capacities, see [19]. Some weaker versions of the conjecture are discussed in [3, 17, 16].




The aim of this note is to study the Erdős-Rado sunflower conjecture for families of bounded dimension. Apart from set systems realized in low-dimensional Euclidean spaces, we consider two additional notions of dimension: the *Vapnik-Chervonenkis dimension* (in short, VC-dimension) and the *Littlestone dimension* (LS-dimension), introduced in [24] and [15], respectively. Both are important combinatorial parameters that measure the complexity of graphs and hypergraphs, and play important roles in statistics, algebraic geometry, PAC learning, and in model theory. There is a growing body of results in extremal combinatorics and Ramsey theory which give much better bounds or stronger conclusions under the additional assumption of bounded dimension (see [12, 13]).

Given a family of sets $\mathcal{F}$ with ground set V, the *VC-dimension* of $\mathcal{F}$, denoted by VC-dim($\mathcal{F}$), is the maximum d for which there exists a d-element set $S \subset V$ such that for every subset $B \subset S$, one can find a member $A \in \mathcal{F}$ with $A \cap S = B$. In this case, we say that S is *shattered* by $\mathcal{F}$.

Let $f_r^d(k)$ denote the least positive integer m such that every family $\mathcal{F}$ of k-sets with $|\mathcal{F}| \geq m$ and VC-dim($\mathcal{F}$) $\leq d$ contains an r-sunflower. Clearly, we have $f_r^d(k) \leq f_r(k)$, and the Erdős-Rado sunflower conjecture implies the following weaker conjecture.

▶ **Conjecture 1.** *For $d \geq 1$ and $r \geq 3$, there is a constant $C = C(d, r)$ such that $f_r^d(k) \leq C^k$.*

It is not difficult to see that, even for $d = 1$, the function $f_r^1(k)$ grows at least exponentially in k. More precisely, we have $f_r^1(k) > (r-1)^{k-1}$. Indeed, consider a rooted complete $(r-1)$-ary tree T with the root on level 0 and with $(r-1)^{k-1}$ leaves on level $k-1$. Let $\mathcal{F}$ be the family of k-sets consisting of the vertex sets of the root-to-leaf paths in T. Obviously, $\mathcal{F}$ does not contain any r-sunflower, and its VC-dimension is at most 1.

More generally, we have the recursive lower bound

$$f_r^d(k_1 + k_2) > (f_r^d(k_1) - 1)(f_r^d(k_2) - 1).$$

Indeed, for $i = 1, 2$, let $\mathcal{F}_i$ be a family of k_i-sets of size $f_r^d(k_i) - 1$ with VC-dimension at most d and without any r-sunflower. For each set S in $\mathcal{F}_1$, make a new copy of $\mathcal{F}_2$ and add S to each set in $\mathcal{F}_2$. The ground set of copies of $\mathcal{F}_2$ are pairwise disjoint for distinct sets of $\mathcal{F}_1$. The resulting set system $\mathcal{F}$ is $(k_1 + k_2)$-uniform with size $(f_r^d(k_1) - 1)(f_r^d(k_2) - 1)$, VC-dimension at most d, and has no r-sunflower. This implies that if $f_r(k') > C^{k'} + 1$ for some k' and C, then there is d depending on k' such that for all sufficiently large k, $f_r^d(k) > C^k$. Thus, any exponential lower bound for the classical sunflower problem (with unbounded VC-dimension) can be achieved by a construction with bounded (but sufficiently large) VC-dimension.

Using a result of Ding, Seymour, and Winkler [10], we settle Conjecture 1 for families of k-sets with VC-dimension $d = 1$.

▶ **Theorem 2.** *For integers $r \geq 3$ and $k \geq 1$, every family of k-sets with VC-dimension $d = 1$ and cardinality at least r^{10k} has an r-sunflower. That is, we have*

$$f_r^1(k) \leq r^{10k}.$$

Let $\log^* k$ denote the *iterated logarithm* of k, *i.e.*, the minimum i for which the i times iterated logarithm of k satisfies $\log^{(i)} k \leq 2$. All logarithms used in this note are of base 2.

For $d \geq 2$, our upper bound on $f_r^d(k)$ is not far from the one stated in Conjecture 1.

▶ **Theorem 3.** *For integers $d, k, r \geq 2$, every family of k-sets with VC-dimension at most d and cardinality at least $2^{10k(dr)^{2\log^* k}}$ has an r-sunflower. In notation,*

$$f_r^d(k) \leq 2^{10k(dr)^{2\log^* k}}.$$

The *Littlestone dimension* of $\mathcal{F} \subseteq 2^V$ is defined as follows. Consider a rooted complete binary tree T_d, with the root at level 0 and with 2^d leaves at the last level. Let the leaves of T_d be labeled by sets in $\mathcal{F}$, and all other vertices by elements of V. We say that T_d is *shattered* by $\mathcal{F}$ if for every root-to-leaf path with labels $v_0, v_1, \ldots, v_{d-1}, F$, we have $v_i \in F$ if and only if the $(i+1)$st vertex along the path is the left-child of v_i, for all $0 \leq i < d$. The Littlestone dimension of $\mathcal{F}$, denoted by LS-dim$(\mathcal{F})$, is the largest d for which there is a labeling of T_d which is shattered by $\mathcal{F}$.

Obviously, we have VC-dim$(\mathcal{F}) \leq$ LS-dim$(\mathcal{F})$, because if the $S = \{s_0, \ldots, s_{d-1}\} \subseteq V$ is shattered by $\mathcal{F}$, then the labeling of T_d in which all vertices at level i are labeled by s_i, $0 \leq s_i < d$, and the leaves by the corresponding sets in $\mathcal{F}$ with the appropriate intersection with S, is also shattered by $\mathcal{F}$.

Let $h_r^d(k)$ denote the least positive integer m such that every family $\mathcal{F}$ of k-sets with $|\mathcal{F}| \geq m$ and LS-dim$(\mathcal{F}) \leq d$ contains an r-sunflower. Since the Littlestone dimension of a set system is at least as large as its VC-dimension, we have

$$h_r^d(k) \leq f_r^d(k) \leq f_r(k).$$

It turns out that $h_r^d(k)$, as a function of k, grows much more slowly than $f_r^d(k)$. Its growth rate is only polynomial in k, albeit the degree of this polynomial depends on d.

▶ **Theorem 4.** *For positive integers d, r, k, every family of k-sets with LS-dimension at most d and cardinality at least $(rk)^d$ has an r-sunflower. Using our notation, we have*

$$h_r^d(k) \leq (rk)^d.$$

On the other hand, for integers $d, r \geq 3$, and $k \geq 4d$, we have

$$h_r^d(k) \geq (rk/d)^{d-o(d)},$$

where the $o(d)$ term goes to 0 as $d \to \infty$.

For several geometrically defined set systems, one can verify the sunflower conjecture by exploring the special properties of the underlying configurations.

A collection $\mathcal{D}$ of Jordan regions in the plane is called a family of *pseudo-disks* if the boundaries of any two members in $\mathcal{D}$ intersect in at most two points. For simplicity, we will assume that $\mathcal{D}$ is in *general position*, that is, no point lies on the boundary of three regions and no two regions are tangent. It is well known that the VC-dimension of the set system obtained by restricting $\mathcal{D}$ to V is at most 3 (see [7]) and, hence, Theorem 3 applies. However, in this case, we can verify the sunflower conjecture.

▶ **Theorem 5.** *Let V be a planar point set and let $\mathcal{D} = \{D_1, \ldots, D_N\}$ be a family of pseudo-disks such that the size of every set $S_i = D_i \cap V$ is equal to k. If $N \geq (500 + r)^{900k}$, where $r > 2$, then there are r distinct sets $S_{i_1}, \ldots, S_{i_r}$ that form an r-sunflower.*

Our paper is organized as follows. Sections 2 and 3 contain the proofs of Theorems 2 and 3, respectively. Theorem 4 about set systems of bounded Littlestone dimension is established in Section 4. Section 5 is devoted to low-dimensional geometric instances of the sunflower conjecture, while the last section contains some concluding remarks.

For the clarity of presentation, throughout this paper we make no attempt to optimize the absolute constants occurring in the statements.

2 VC-dimension 1–Proof of Theorem 2

Given a family $\mathcal{F}$ of subsets of a ground set V, as usual, let $\nu(\mathcal{F})$ denote the *packing number* of $\mathcal{F}$, *i.e.,* the maximum number of pairwise disjoint members of $\mathcal{F}$. Also, let $\tau(\mathcal{F})$ be the *transversal number* of $\mathcal{F}$, *i.e.*, the minimum number of elements that can be selected from V such that every member of $\mathcal{F}$ contains at least one of them. Finally, let $\lambda(\mathcal{F})$ denote the maximum integer l such that there are l sets $S_1, \ldots, S_l \in \mathcal{F}$ with the property that for any $1 \leq i < j \leq l$, there is $v = v_{ij} \in S_i \cap S_j$ such that $v \notin S_t$ for $t \in [m] \setminus \{i, j\}$. It is easy to verify that $\lambda(\mathcal{F})$ is at least as large as the VC-dimension of the set system (hypergraph) $\mathcal{F}^*$ *dual* to $\mathcal{F}$.

We need the following result of Ding, Seymour, and Winkler [10] which bounds the transversal number of $\mathcal{F}$ in terms of its packing number and $\lambda(\mathcal{F})$.

▶ **Lemma 6** (Ding, Seymour, Winkler)**.** *Let $\mathcal{F}$ be a set system with ground set V, and let $\nu(\mathcal{F}) = \nu, \tau(\mathcal{F}) = \tau$ and $\lambda(\mathcal{F}) = \lambda$. Then we have*

$$\tau \leq 11\lambda^2(\lambda + \nu + 3)\binom{\lambda + \nu}{\lambda}^2.$$

Notice that VC-dim$(\mathcal{F}) = 1$ implies that $\lambda(\mathcal{F}) \leq 3$. Hence, Theorem 2 is an immediate corollary to the following result.

▶ **Theorem 7.** *Let $r \geq 3$ and let $\mathcal{F}$ be a family of k-sets with $\lambda(\mathcal{F}) = \lambda$ which does not contain an r-sunflower. Then we have $|\mathcal{F}| \leq (\lambda + r)^{6\lambda k}$.*

Proof. We proceed by induction on k. The base case $k = 1$ follows from the trivial bound $|\mathcal{F}| \leq r - 1$. The induction hypothesis is that the bound holds for families of $(k-1)$-sets. For the inductive step, let $\mathcal{F} \subseteq 2^V$ be a family of k-sets with no r-sunflower. In particular, $\mathcal{F}$ has no r disjoint members, so that $\nu(\mathcal{F}) < r$. By Lemma 6,

$$\begin{aligned} \tau(\mathcal{F}) &\leq 11\lambda^2(\lambda + r + 3)\tbinom{\lambda + r}{\lambda}^2 \leq 11\lambda^2(\lambda + r + 3)(\lambda + r)^{2\lambda}(\lambda!)^{-2} \\ &\leq 11(\lambda + r + 3)(\lambda + r)^{2\lambda} \leq 20(\lambda + r)^{2\lambda + 1}. \end{aligned}$$

Therefore, there is $v \in V$ incident to at least $|\mathcal{F}|/\tau(\mathcal{F}) \geq |\mathcal{F}|/\left(20(\lambda + r)^{2\lambda+1}\right)$ members of $\mathcal{F}$.

Let $\mathcal{F}' = \{S \setminus \{v\} : S \in \mathcal{F}, v \in S\}$. Then we have $|\mathcal{F}'| \geq |\mathcal{F}|/\left(20(\lambda + r)^{2\lambda+1}\right)$, $\lambda(\mathcal{F}') \leq \lambda(\mathcal{F})$, and $\mathcal{F}'$ does not contain any r-sunflower. By the induction hypothesis, we have $|\mathcal{F}'| \leq (\lambda + r)^{6\lambda(k-1)}$. Thus, we obtain

$$|\mathcal{F}| \leq 20(\lambda + r)^{2\lambda+1}|\mathcal{F}'| \leq 20(\lambda + r)^{2\lambda+1}(\lambda + r)^{6\lambda(k-1)} \leq (\lambda + r)^{6\lambda k},$$

as required. ◀

3 Bounded VC-dimension–Proof of Theorem 3

In this section, we prove Theorem 3, which is the main result of this paper. We need the following lemma due to Sauer [20], Shelah [22], Perles, and, in a slightly weaker form, to Vapnik and Chervonenkis [24]. See also [18].

▶ **Lemma 8** (Sauer, Shelah, Perles)**.** *Let $\mathcal{F}$ be a set system with ground set V and VC-dimension at most d. Then we have $|\mathcal{F}| \leq \sum_{i=0}^{d} \binom{|V|}{i}$.*

Before turning to the proof, we need to discuss some closely related variants of the sunflower problem.

First, we could ask the same question for *multifamilies* of sets, that is, for collections of not necessarily distinct sets. Let $g_r(k)$ be the minimum positive integer m such that every multifamily of k-sets of size m contains an r-sunflower. It is an easy exercise to prove that $g_r(k) = (r-1)f_r(k) + 1$.

Analogously, for any $d \geq 1$, let $g_r^d(k)$ be the minimum positive integer m such that every multifamily of k-sets of size m with *VC-dimension at most* d contains an r-sunflower. We similarly have $g_r^d(k) = (r-1)f_r^d(k) + 1$.

To obtain upper bounds for $g_r^d(k)$ and $f_r^d(k)$, we define the following related function. Let $\alpha_r^d(k)$ denote the maximum α such that for every nonempty multifamily $\mathcal{F}$ of k-sets with VC-dimension at most d, if we select r members uniformly at random from $\mathcal{F}$ with replacement, the probability that they have pairwise equal intersections is at least α.

Next, notice that the value of $f_r(k)$ remains the same if we change the definition from families of k-sets to families of sets with *at most* k elements. Indeed, this can be achieved by adding distinct "dummy" vertices to each set of size smaller than k so that it will have size exactly k. The same holds for the functions $f_r^d(k), g_r(k)$, $g_r^d(k)$, and $\alpha_r^d(k)$ because adding dummy vertices does not affect the VC-dimension of the family.

Considering a family of VC-dimension d which consists of $f_r^d(k) - 1$ sets of size k and contains no r-sunflower, we immediately obtain the following upper bound on $\alpha_r^d(k)$ as the r-tuples of sets from the family that have pairwise equal intersections are those that consist of the same set r times.

$$\alpha_r^d(k) \leq (f_r^d(k) - 1)^{1-r}. \tag{1}$$

The following lemma implies that this bound on $\alpha_r^d(k)$ is tight within a factor er^{r-1}.

▶ **Lemma 9.** *For integers $d, k, r \geq 2$ we have*

$$\alpha_r^d(k) \geq g_r^d(k)^{1-r}/e.$$

Proof. Let $S_r^d(m,k)$ denote the minimum possible number of r-sunflowers in a multifamily $\mathcal{F}$ of at most k-element sets with cardinality m and VC-dimension at most d. From the definition, if $m < g_r^d(k)$, then $S_r^d(m,k) = 0$, while if $m \geq g_r^d(k)$, then $S_r^d(m,k) \geq 1$.

Our argument is based on the proof technique used to obtain the "crossing lemma" [2], see also [23]. The idea is to use an averaging (or, equivalently, probabilistic) argument to amplify a weak bound to a better bound. By deleting one set from each r-sunflower, we get the trivial bound $S_r^d(m,k) \geq m - g_r^d(k) + 1$. For $M \geq m$, by averaging over all subfamilies of size m, we obtain

$$S_r^d(M,k) \geq S_r^d(m,k)\binom{M}{r}/\binom{m}{r}.$$

In particular, $S_r^d(m,k)/\binom{m}{r}$ is a monotone increasing function of m. Set $m_0 = (1+1/r)g_r^d(k) - 1$. Then we have $S_r^d(m_0,k) \geq m_0 - g_r^d(k) + 1 = g_r^d(k)/r$. Thus, for $m \geq m_0$, we have

$$\begin{aligned} S_r^d(m,k) &\geq S_r^d(m_0,k)\binom{m}{r}/\binom{m_0}{r} \\ &\geq \frac{1}{r}g_r^d(k)\binom{m}{r}/\binom{(1+1/r)g_r^d(k)}{r} \\ &\geq \frac{1}{er}g_r^d(k)^{1-r}m^r. \end{aligned} \tag{2}$$

Let $\alpha_r^d(m,k)$ be the maximum α with the property that for every multifamily $\mathcal{F}$ of at most k-element sets with cardinality m and VC-dimension at most d, if we uniformly at random choose r sets from $\mathcal{F}$ with replacement, the probability that they have pairwise equal intersections is at least α. Thus,

$$\alpha_r^d(m,k) \geq S_r^d(m,k)/\binom{m}{r} + m^{1-r}, \tag{3}$$

where the first term comes from possibly choosing r different sets (in terms of label, if we view the m not necessarily distinct sets as labeled from 1 to m), and the second term comes from possibly choosing the same set r times.

For $m \geq m_0$, by using (3) and then (2), we have

$$\alpha_r^d(m,k) \geq S_r^d(m,k)/\binom{m}{r} \geq r!S_r^d(m,k)m^{-r} \geq (r-1)!g_r^d(k)^{1-r}/e.$$

For $m < m_0$, using the trivial bound $S_r^d(m,k) \geq 0$, we have

$$\alpha_r^d(m,k) \geq m^{1-r} > m_0^{1-r} = \left((1+1/r)g_r^d(k) - 1\right)^{1-r} \geq g_r^d(k)^{1-r}/e.$$

As $\alpha_r^d(k) = \inf_m \alpha_r^d(m,k)$, we have the desired bound $\alpha_r^d(k) \geq g_r^d(k)^{1-r}/e$. ◀

Combining the previous lemma with the Erdős-Rado bound $f_r(k) \leq k!(r-1)^k$, and the inequality $g_r^d(k) \leq g_r(k) = (k-1)f_r(k)+1$, we obtain the following corollary.

▶ **Corollary 10.** *For any integers $d, k, r \geq 2$, we have*

$$\alpha_r^d(k) \geq \left(k!(r-1)^{k+1}+1\right)^{1-r}/e.$$

We are now in a position to prove the following result which, together with (1), immediately implies Theorem 3.

▶ **Theorem 11.** *For any $d, k, r \geq 2$, we have*

$$\alpha_r^d(k) \geq 2^{-10k(dr)^{2\log^* k}}.$$

Proof. If $r = 2$, then we have $\alpha_2^d(k) = 1$ and the result follows. Therefore we can assume $r \geq 3$. We use induction on k. For the base cases $k < 8$, by Corollary 10, we have

$$\alpha_r^d(k) \geq \left(k!(r-1)^{k+1}+1\right)^{1-r}/e \geq 2^{-10k(dr)^{2\log^* k}}.$$

For the inductive step, let $k \geq 8$ and assume that the statement holds for all $k' < k$. Let $\mathcal{F}$ be a non-empty multifamily of at most k-element sets with VC-dimension at most d. Without loss of generality, we may assume that the ground set is $\mathbb{N}$. Let ϵ_i be the fraction of sets in $\mathcal{F}$ that contain i. By reordering the elements of the ground set, if necessary, we may also assume that $\epsilon_1 \geq \epsilon_2 \geq \ldots$, that is, the elements of the ground set are ordered in decreasing frequency.

As each member of $\mathcal{F}$ has size at most k, the expected size of the intersection of $[s] = \{1, 2, \ldots, s\}$ with a randomly selected member of $\mathcal{F}$ is at most k. On the other hand, this expectation is $\epsilon_1 + \cdots + \epsilon_s \geq s\epsilon_s$. Therefore, we have $\epsilon_s \leq k/s$.

Set $s = \lceil 4k^4/\alpha_r^d(\log k)\rceil$. Define two multifamilies, $\mathcal{F}_1$ and $\mathcal{F}_2$, as follows. Let

$$\mathcal{F}_1 = \{S : S \in \mathcal{F} \text{ and } |S \cap [s]| \leq \log k\}, \quad \mathcal{F}_2 = \mathcal{F} \setminus \mathcal{F}_1.$$

Thus, we have $|\mathcal{F}| = |\mathcal{F}_1| + |\mathcal{F}_2|$. We select at random, uniformly and independently with repetition, r sets $S_1, \dots, S_r \in \mathcal{F}$. Let X denote the event that the r sets form an r-sunflower. The proof now falls into two cases.

Case 1: Suppose that $|\mathcal{F}_1| \geq (1 - 1/r)|\mathcal{F}|$. Let Y denote the event that $S_1, \dots, S_r \in \mathcal{F}_1$. Let Z be the event that $S_1 \cap [s], \dots, S_r \cap [s]$ have pairwise equal intersections, and let W be the event that $S_1 \setminus [s], \dots, S_r \setminus [s]$ are pairwise disjoint. Hence,

$$\mathbb{P}[X] \geq \mathbb{P}[Y \cap Z \cap W] = \mathbb{P}[Y \cap Z] - \mathbb{P}[Y \cap Z \cap \bar{W}] \geq \mathbb{P}[Y \cap Z] - \mathbb{P}[\bar{W}]. \tag{4}$$

Clearly, we have

$$\mathbb{P}[Y] \geq (1 - 1/r)^r \geq \frac{1}{4}, \tag{5}$$

and, by definition,

$$\mathbb{P}[Z \mid Y] \geq \alpha_r^d(\log k). \tag{6}$$

Therefore, by (5) and (6), we have

$$\mathbb{P}[Y \cap Z] = \mathbb{P}[Y]\mathbb{P}[Z \mid Y] \geq \frac{1}{4}\alpha_r^d(\log k). \tag{7}$$

Fixing $S_i \setminus [s]$, which has size at most k, the probability that $S_j \setminus [s]$ contains at least one of the elements of $S_i \setminus [s]$ is at most $k\epsilon_{s+1} \leq k^2/(s+1)$. Hence, by the probability union bound, we have

$$\mathbb{P}[\bar{W}] \leq \frac{\binom{k}{2}k^2}{s+1} < \frac{k^4}{2s} \leq \frac{\alpha_r^d(\log k)}{8}. \tag{8}$$

Combining (4), (7), and (8), we obtain

$$\mathbb{P}[X] = \mathbb{P}[Y \cap Z] - \mathbb{P}[\bar{W}] \geq \frac{\alpha_r^d(\log k)}{4} - \frac{\alpha_r^d(\log k)}{8} = \frac{\alpha_r^d(\log k)}{8}.$$

Hence, by the induction hypothesis, we have

$$\alpha_r^d(k) \geq \mathbb{P}[X] \geq \frac{1}{8}\alpha_r^d(\log k) \geq \frac{1}{8}2^{-10(\log k)(dr)^{2\log^* k - 2}} \geq 2^{-10k(dr)^{2\log^* k}}.$$

Case 2: Suppose that $|\mathcal{F}_2| \geq |\mathcal{F}|/r$. Since $\mathcal{F}$ has VC-dimension at most d, by the Sauer-Shelah-Perles lemma, Lemma 8, the number of distinct sets in $\{S \cap [s] : S \in \mathcal{F}\}$ is at most s^d. By the pigeonhole principle, there is a subset $A \subset [s]$ with $|A| \geq \log k$ such that the family

$$\mathcal{F}' = \{S \in \mathcal{F} : S \cap [s] = A\}$$

has at least $|\mathcal{F}_2|/s^d \geq |\mathcal{F}|/(rs^d)$ members.

Select r sets $S_1, \dots, S_r$ from $\mathcal{F}$ uniformly at random with repetition. Let Y' denote the event that $S_1, \dots, S_r \in \mathcal{F}'$ and let Z' denote the event that $S_1 \setminus [s], \dots, S_r \setminus [s]$ form an r-sunflower. Hence,

$$\begin{aligned}
\mathbb{P}[X] &\geq \mathbb{P}[Y' \cap Z'] \\
&= \mathbb{P}[Y'] \cdot \mathbb{P}[Z' \mid Y'] \\
&\geq \left(\tfrac{1}{rs^d}\right)^r \alpha_r^d(k - \log k) \\
&\geq \tfrac{1}{r^r(5k^4)^{dr}} \left(\alpha_r^d(\log k)\right)^{dr} \alpha_r^d(k - \log k).
\end{aligned}$$

By the induction hypothesis, we obtain

$$\mathbb{P}[X] \geq \frac{1}{r^r(5k^4)^{dr}} \left(2^{-10(\log k)(dr)^{2\log^* k - 2}}\right)^{dr} \left(2^{-10(k-\log k)(dr)^{2\log^* k}}\right).$$

Since $dr \geq 6$ and $k \geq 8$, we have

$$\mathbb{P}[X] \geq \frac{1}{r^r(5k^4)^{dr}} 2^{-10k(dr)^{2\log^* k} + 8\log k(dr)^{2\log^* k}} \geq 2^{-10k(dr)^{2\log^* k}}.$$

This completes the proof. ◀

4 Littlestone dimension – Proof of Theorem 4

Originally, the Littlestone dimension was introduced for the characterization of regret bounds in online learning, see [4, 15, 6]. As Chase and Freitag [8] pointed out, the notion is equivalent to Shelah's model theoretic rank. The definition can also be reformulated as follows.

For a finite family $\mathcal{F}$ of sets with ground set V, define LS-dim($\mathcal{F}$), the *Littlestone dimension* of $\mathcal{F}$, recursively. If $|\mathcal{F}| \leq 1$, then let LS-dim($\mathcal{F}$) $= 0$. For an element x of the ground set, let $\mathcal{F}_x = \{S \setminus \{x\} : x \in S \text{ and } S \in \mathcal{F}\}$ and $\mathcal{F}'_x = \{S : x \notin S \text{ and } S \in \mathcal{F}\}$. If $|\mathcal{F}| > 1$, then let

$$\text{LS-dim}(\mathcal{F}) = 1 + \max_{x \in V} \min\left(\text{LS-dim}(\mathcal{F}_x), \text{LS-dim}(\mathcal{F}'_x)\right).$$

For $d \geq 1$, let $h^d_r(k)$ be the minimum positive integer m such that every family of k-sets with size at least m and Littlestone dimension at most d contains an r-sunflower.

▶ **Lemma 12.** *For positive integers k and r, we have $h^1_r(k) = k + r - 1$.*

Proof. We have $h^1_r(k) > k + r - 2$ by considering the following family $\mathcal{F}_{r,k}$ of k-sets. For $k = 1$, let the family consist of $r - 1$ singleton sets. For $k > 1$, we obtain $\mathcal{F}_{r,k}$ from $\mathcal{F}_{r,k-1}$ by adding one new ground element to all sets in $\mathcal{F}_{r,k-1}$, and then including one additional k-set with entirely new ground elements. It is straightforward to check that this family of k-sets has $k + r - 2$ members, its Littlestone dimension is 1, and it does not contain any r-sunflower.

We prove the upper bound inductively on k, with the base case $k = 1$ being trivial. Let $k \geq 2$ and let $\mathcal{F}$ be a family of k-sets with size $h^1_r(k) - 1$ which has Littlestone dimension at most 1 and does not contain an r-sunflower. A family of sets has Littlestone dimension at most 1 if and only if every element x of the ground set belongs to *at most one* or to *all but at most one* set in the family, that is, if $|\mathcal{F}_x| \leq 1$ or $|\mathcal{F}'_x| \leq 1$ for all x. If there is an element x for which $|\mathcal{F}'_x| \leq 1$, then $|\mathcal{F}_x| = |\mathcal{F}| - 1 = h^1_r(k) - 2$ and $\mathcal{F}_x$ is a family of $(k-1)$-sets of Littlestone dimension at most 1 which does not contain an r-sunflower, from which we obtain $h^1_r(k-1) \leq h^1_r(k-1) + 1$. If there is no ground element x in more than one set in $\mathcal{F}$, then all members of $\mathcal{F}$ are disjoint. Therefore, $|\mathcal{F}| < r$ and $h^1_r(k) \leq r$. ◀

▶ **Lemma 13.** *For any family $\mathcal{F}$ of sets of size at most k with no $(r+1)$-sunflower, there is an element of the ground set which belongs to at least a $\frac{1}{kr}$-fraction of the sets.*

Proof. Consider a maximum family $\{S_1, \ldots, S_s\}$ of sets in $\mathcal{F}$ which are pairwise disjoint. Such a family forms a sunflower and hence $s \leq r$. In particular, any set in $\mathcal{F}$ contains at least one element from $\bigcup_{i=1}^{s} S_i$, which has a total of $ks \leq kr$ elements. By the pigeonhole principle, there is an element of the ground set which belongs to at least a fraction $\frac{1}{kr}$ of the sets in $\mathcal{F}$. ◀

▶ **Lemma 14.** *For integers $k, r \geq 1$ and $d \geq 2$, we have*

$$h_r^d(k) \leq \max\left(k(r-1)\left(h_r^{d-1}(k-1)-1\right)+1, h_r^d(k-1)+h_r^{d-1}(k)-1\right).$$

Proof. Let $\mathcal{F}$ be a family of k-sets with size $h_r^d(k)-1$ which has Littlestone dimension at most d and does not contain an r-sunflower. By Lemma 13, there is an element x of the ground set in at least a fraction $\frac{1}{k(r-1)}$ of the sets in $\mathcal{F}$. As $\mathcal{F}$ has Littlestone dimension d, at least one of $\mathcal{F}_x$ or $\mathcal{F}'_x$ has Littlestone dimension at most $d-1$.

If $\mathcal{F}_x$ has Littlestone dimension at most $d-1$, then $\mathcal{F}_x$ is a family of $(k-1)$-sets which has no r-sunflower, and hence

$$\frac{1}{k(r-1)}\left(h_r^d(k)-1\right) = \frac{1}{k(r-1)}|\mathcal{F}| \leq |\mathcal{F}_x| \leq h_r^{d-1}(k-1)-1,$$

from which it follows that $h_r^d(k) \leq k(r-1)\left(h_r^{d-1}(k-1)-1\right)+1$.

If $\mathcal{F}'_x$ has Littlestone dimension at most $d-1$, then we have $|\mathcal{F}'_x| \leq h_r^{d-1}(k)-1$ and $|\mathcal{F}'_x| \leq h_r^d(k-1)-1$, from which it follows that

$$h_r^d(k)-1 = |\mathcal{F}| = |\mathcal{F}_x| + |\mathcal{F}'_x| \leq h_r^{d-1}(k)-1+h_r^d(k-1)-1,$$

and, hence, $h_r^d(k) \leq h_r^{d-1}(k)+h_r^d(k-1)-1$. ◀

We can now prove Theorem 4.

Proof of Theorem 4. For the upper bound, the proof is by induction on the Littlestone dimension d. In the base case $d=1$, we have $h_r^1(k) = k+r-1 \leq kr$. Suppose $d \geq 2$. Consider the recursive upper bound on $h_r^d(k)$ from Lemma 14. We split the proof into two cases depending on the maximum of the two functions in the upper bound on $h_r^d(k)$. In each case, we use the induction hypothesis.

In the first case, we have

$$h_r^d(k) \leq k(r-1)\left(h_r^{d-1}(k-1)-1\right)+1 \leq kr(kr)^{d-1} = (kr)^d.$$

In the latter case, we have

$$h_r^d(k) \leq h_r^{d-1}(k)+h_r^d(k-1)-1 < (kr)^{d-1}+((k-1)r)^d \leq (kr)^{d-1}+(1-\frac{1}{k})(kr)^d < (kr)^d.$$

In either case, we obtained the desired bound.

For the lower bound, let $d \geq 6$, $r \geq 3$, k be sufficiently large with $k \geq 4d$, $n = k^2 r/(500d\log k)$, $t = \lceil \log d \rceil$ and $m = n^{-1}(n/k)^{d-t}$. We use the probabilistic method to show that there exists a family $\mathcal{F}$ of k-element subsets of $[n] := \{1,\ldots,n\}$ with $|\mathcal{F}| \geq m/2$, $\mathcal{F}$ does not contain any r-sunflower, and the Littlestone dimension of $\mathcal{F}$ is at most d. This implies the desired lower bound on $h_r^d(k)$.

We will show that $\mathcal{F}$ satisfies four properties each with high probability. This means that the probability is of the form $1-o(1)$ with the $o(1)$ term tending to 0 as k tends to infinity. Hence, all four properties hold with high probability. These four properties guarantee that $\mathcal{F}$ has the desired properties and hence there is a choice of $\mathcal{F}$ with the desired properties.

Pick m subsets $S_1,\ldots,S_m \subset \binom{[n]}{k}$ uniformly and independently at random. Let $\mathcal{F}$ be the family of distinct S_i. Since $m \ll \binom{n}{k}$, it is easy to see that, with high probability, we have $|\mathcal{F}| \geq m/2$.

We next show that, with high probability, $\mathcal{F}$ does not contain any r-sunflower. Consider a subsequence of r of these random sets, say $S_1, \ldots, S_r$. The number of sequences of r sets in $\binom{[n]}{k}$ which have pairwise equal intersection, and this intersection has size s, is

$$\binom{n}{s} \prod_{i=1}^{r} \binom{n-s-(i-1)(k-s)}{k-s} = s!^{-1}(k-s)!^{-r} n! / (n-s-r(k-s))!$$

This is because there are $\binom{n}{s}$ ways of choosing the common intersection of size s, and given the S_j with $j < i$, the remaining $k-s$ elements from S_i not in the common intersection must be chosen from the $n-s-(i-1)(k-s)$ elements not in any of the S_j with $j < i$. As there are $\binom{n}{k}$ ways to pick each S_i, the probability that the r random sets $S_1, \ldots, S_r$ have pairwise equal intersection of size s is

$$\binom{n}{k}^{-r} s!^{-1}(k-s)!^{-r} \frac{n!}{(n-s-r(k-s))!} = \binom{k}{s} \cdot (k!/(k-s)!)^{r-1} \cdot \frac{n!/(n-s-r(k-s))!}{\left(k!\binom{n}{k}\right)^{r}}. \quad (9)$$

Note that the expression in the right hand side of (9) is the product of three factors. The middle factor is at most $k^{s(r-1)}$. In the third factor in the right hand side of (9), the numerator can be expressed as the product of factors $(n-j)$ for $j = 0, \ldots, s + r(k-s) - 1$, which is a total of $s + r(k-s)$ factors, while the denominator can be expressed as the product of rk factors which are of the form $(n-h)$ with $h \leq k$. It follows that the third factor in the right hand side of (9) is at most

$$(n-k)^{-s(r-1)} \prod_{j=1}^{(r-1)(k-s)-1} \left(1 - \frac{j}{n-k}\right) \leq (n-k)^{-s(r-1)} e^{-(r-1)^2(k-s)^2/(4n)},$$

where we used the inequality $1 - x \leq e^{-x}$ for $x \geq 0$ to bound each factor in the product.

It follows that the expression on the right hand side of (9) is at most

$$\left(k\,(k/(n-k))^{r-1}\right)^{s} \cdot e^{-(r-1)^2(k-s)^2/(4n)}. \quad (10)$$

Thus, the probability that $S_1, \ldots, S_r$ form an r-sunflower is at most

$$\sum_{s=0}^{k} \left(k\,(k/(n-k))^{r-1}\right)^{s} e^{-(r-1)^2(k-s)^2/(4n)}.$$

We bound the probability that there is an r-sunflower in $\mathcal{F}$ by taking the union bound over all the $\binom{m}{r}$ choices of r sets from $S_1, \ldots, S_m$. Note that (10) is the product of two factors which are each at most 1. We bound (10) for $s \geq 2d$ by the first factor, and for $s < 2d$ by the second factor. We also use the inequality $\binom{m}{r} \leq (em/r)^r$. Substituting in the chosen values for n and m, we get a $o(1)$ probability that there is an r-sunflower in $\mathcal{F}$.

Finally, we bound the probability that $\mathcal{F}$ has Littlestone dimension greater than d. If $\mathcal{F}$ has Littlestone dimension greater than d, in the rooted complete binary tree realizing the Littlestone dimension, going down from the root by taking the left-child each time for $d-t$ levels, we see that there are at least $2^t \geq d$ sets that each contain the same $d-t$ vertices. So the probability that $\mathcal{F}$ has Littlestone dimension greater than d is at most the probability that there are d sets in $\mathcal{F}$ that each contain the same $d-t$ elements from $[n]$. This probability in turn is at most

$$\binom{m}{d}\binom{n}{d-t}\left(\frac{k}{n}\right)^{(d-t)d} < \left(\frac{em}{d}\right)^{d}\left(\frac{en}{d-t}\right)^{d-t}\left(\frac{k}{n}\right)^{(d-t)d} = o(1).$$

Here we used the union bound over all $\binom{m}{d}$ choices of d indices from $[m]$ and over all $\binom{n}{d-t}$ choices of $d-t$ distinct integers in $[n]$. We also used that the probability that a given set of $d-t$ elements in $[n]$ is in a random k-set is at most $(k/n)^{d-t}$. The last inequality is by substituting in the chosen values of n and m. ◀

5 Geometric versions of the sunflower conjecture

We start with the proof of Theorem 5. We need the following lemma due to Sharir [21].

▶ **Lemma 15.** *Let $\mathcal{D} = \{D_1, \ldots, D_t\}$ be a family of pseudo-disks in the plane, and let P denote the set of all intersection points of the boundaries of D_i. Then the number of points in P covered by the interior of at most k other regions D_i is at most $26kt$.*

Proof of Theorem 5. Given $r > 2$, let $N = (500 + r)^{900k}$. We proceed by induction on k. The base case $k = 1$ is trivial. Now assume the statement holds for $k' < k$.

Let V be a planar point set and let $\mathcal{D} = \{D_1, \ldots, D_N\}$ be a family of pseudo-disks in the plane such that $|D_i \cap V| = k$ for all i. By slightly perturbing each region D_i, we can assume that no point in V lies on the boundary of D_i for all i. Set $S_i = D_i \cap V$ and $\mathcal{F} = \{S_1, \ldots, S_N\}$.

Let $t = \lambda(\mathcal{F})$ and suppose the sets $S_1, \ldots, S_t \in \mathcal{F}$ have the property that for any $1 \le i < j \le t$, there is a vertex $v \in S_i \cap S_j$ from V such that $v \notin S_\ell$ for $\ell \in [t] \setminus \{i, j\}$. Then, by letting C_i denote the boundary of D_i, there are at least $\binom{t}{2}$ connected components in $\mathbb{R}^2 \setminus \bigcup_i C_i$ that are covered by at most two regions D_i. On the other hand, by Lemma 15, there are at most $4(52)t$ such regions, since every point in the arrangement $\bigcup_i C_i$ is incident to at most four such connected components. Therefore, we have

$$\binom{t}{2} \le 208t,$$

which implies that $t \le 417$.

Further, we can assume that $\nu(\mathcal{F}) \le r - 1$, since otherwise we would be done. By Lemma 6, we have

$$\tau(\mathcal{F}) \le 11(417)^2(419 + r)\binom{416 + r}{417}^2 \le (500 + r)^{900}.$$

There is a vertex $v \in V$ which is incident to at least $N/\tau(\mathcal{F})$ members in $\mathcal{F}$. Let $\mathcal{D}' = \{D_i \in \mathcal{D} : v \in D_i\}$, $V' = V \setminus \{v\}$, and $S_i' = V' \cap D_i'$. Hence, $|\mathcal{D}'| \ge N/\tau(\mathcal{F}) \ge (500 + r)^{900(k-1)}$. By the induction hypothesis, there are r sets $S_{i_1}', \ldots, S_{i_r}'$ in $\mathcal{D}'$ that form an r-sunflower. Together with vertex v, we obtain an r-sunflower in $\mathcal{F}$. ◀

Replacing Lemma 15 with Clarkson's theorem on levels in arrangement of hyperplanes [9], the argument above gives the following.

▶ **Theorem 16.** *Given $r > 2$, there is a constant $C = C(r)$ for which the following statement is true. If V is a point set in $\mathbb{R}^3$ and $\mathcal{H} = \{H_1, \ldots, H_N\}$ is a family of $N \ge C^k$ half-spaces such that the size of the set $S_i = H_i \cap V$ is k for all i, then there are r distinct sets $S_{i_1}, \ldots, S_{i_r}$ that form an r-sunflower.*

6 Concluding Remarks

The Erdős-Rado sunflower conjecture remains an outstanding open problem. Although we made progress in this paper, it still remains open for families of bounded VC-dimension.

We were able to prove the conjecture in a geometric setting in the plane (Theorem 5). We think it would be interesting to prove the conjecture in other geometric settings, such as for families of sets that are the intersection of the ground set with semi-algebraic sets of bounded description complexity. Such families are of bounded VC-dimension. The following conjecture is a natural special case to consider.

▶ **Conjecture 17.** *For each integer $r \geq 3$, there is a constant $C = C(r)$ such that the following holds. If $V \subset \mathbb{R}^3$ and $\mathcal{F}$ is a family of subsets of V each of size k with $|\mathcal{F}| \geq C^k$ such that every set in $\mathcal{F}$ is the intersection of V with a unit ball in $\mathbb{R}^3$, then $\mathcal{F}$ contains an r-sunflower.*

References

1 H. L. Abbott, D. Hanson, and N. Sauer. Intersection theorems for systems of sets. *J. Combin. Theory Ser. A*, 12:381–389, 1972.

2 M. Ajtai, V. Chvátal, M.M. Newborn, and E. Szemerédi. Crossing-free subgraphs. In Peter L. Hammer, Alexander Rosa, Gert Sabidussi, and Jean Turgeon, editors, *Theory and Practice of Combinatorics*, volume 60 of *North-Holland Mathematics Studies*, pages 9–12. North-Holland, 1982. doi:10.1016/S0304-0208(08)73484-4.

3 N. Alon and R. Holzman. Near-sunflowers and focal families. *arXiv*, 2020. arXiv:2010.05992.

4 N. Alon, R. Livni, M. Malliaris, and S. Moran. Private pac learning implies finite littlestone dimension. *STOC*, pages 852–860, 2019.

5 R. Alweiss, S. Lovet, K. Wu, and J. Zhang. Improved bounds for the sunflower lemma. *arXiv*, 2020. arXiv:1908.08483.

6 S. Ben-David, D. Pál, and S. Shalev-Shwartz. Agnostic online learning. *COLT*, 2009.

7 S. Buzaglo, R. Holzman, and R. Pinchasi. On s-intersecting curves and related problems. *Symposium on Computational Geometry*, pages 79–84, 2008.

8 H. Chase and J. Freitag. Model theory and machine learning. *Bull. Symb. Log.*, 25:319–332, 2019.

9 K. L. Clarkson. Applications of random sampling in computational geometry, ii. *Symposium on Computational Geometry*, pages 1–11, 1988.

10 G. Ding, P. Seymour, and P. Winkler. Bounding the vertex cover number of a hypergraph. *Combinatorica*, 14:23–34, 1994.

11 P. Erdős and R. Rado. Intersection theorems for systems of sets. *J. London Math. Soc.*, 35:85–90, 1960.

12 J. Fox, J. Pach, and A. Suk. Erdős-Hajnal conjecture for graphs with bounded VC-dimension. *Discrete Comput. Geom.*, 61:809–829, 2019.

13 J. Fox, J. Pach, and A. Suk. Bounded VC-dimension implies the Schur-Erdős conjecture. *Symposium on Computational Geometry*, pages 46:1–46:8, 2020.

14 A. V. Kostochka. An intersection theorem for systems of sets. *Random Structures Algorithms*, 9:213–221, 1996.

15 N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Mach. Learn.*, 2:285–318, 1987.

16 E. Naslund and W. Sawin. Upper bounds for sunflower-free sets. *Forum Math. Sigma*, e15:10 pp., 2017.

17 P. Erdős and E. Szemerédi. Combinatorial properties of systems of sets. *J. Combin. Theory Ser. A*, 24:308–313, 1978.

18 J. Pach and P. Agarwal. *Combinatorial Geometry*. Wiley-Interscience, New York, 1995.

19 A. Rao. Coding for sunflowers. *Discrete Anal.*, 2:8 pp., 2020.

20 N. Sauer. On the density of families of sets. *J. Combinat. Theory Ser. A*, 13:145–147, 1972.

21 M. Sharir. On k-sets in arrangements of curves and surfaces. *Discrete Comput. Geom.*, 6:593–617, 1991.

22 S. Shelah. A combinatorial problem, stability and order for models and theories in infinitary languages. *Pacific J. Math.*, 41:247–261, 1972.

23 S. Smorodinsky and M. Sharir. Selecting points that are heavily covered by pseudo-circles, spheres or rectangles. *Combin. Probab. Comput.*, 13:389–411, 2004.

24 V. Vapnik and A. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory Probab. Appl.*, 16:264–280, 1971.

Strong Hanani-Tutte for the Torus

Radoslav Fulek
Department of Mathematics, UC San Diego, La Jolla, CA, USA

Michael J. Pelsmajer
Department of Applied Mathematics, Illinois Institute of Technology, Chicago, IL, USA

Marcus Schaefer
Department of Computer Science, DePaul University, Chicago, IL, USA

Abstract

If a graph can be drawn on the torus so that every two independent edges cross an even number of times, then the graph can be embedded on the torus.

2012 ACM Subject Classification Mathematics of computing → Graphs and surfaces

Keywords and phrases Graph Embedding, Torus, Hanani-Tutte Theorem, Intersection Form

Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.38

Related Version *Full Version*: https://arxiv.org/abs/2009.01683

Funding *Marcus Schaefer*: This work was supported by a DePaul/CDM Faculty Summer Research Stipend Program.

1 Introduction

Two edges in a graph are *independent* if they do not share a vertex. A drawing of a graph on a surface is *independently even*, or *iocr*-0 for short, if every two independent edges cross[1] an even number of times in the drawing.

In the plane, there is a beautiful characterization of planar graphs known as the Hanani-Tutte theorem which says that a graph has an embedding in the plane (i.e., is planar) if and only if it has an independently even drawing in the plane. Equivalently, any drawing of a non-planar graph in the plane must contain two independent edges that cross oddly. The Hanani-Tutte theorem has a wide array of applications in computational and combinatorial geometry [32], some of which we discuss below.

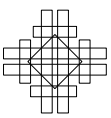
There are several proofs of the Hanani-Tutte theorem, including the original 1934 proof by Hanani and the 1970 proof by Tutte; see [26] for more references. We also know that the result remains true for the projective plane[2] [4, 25]. On the other hand, counterexamples were found recently which show that the Hanani-Tutte theorem does not extend to orientable surfaces of genus 4 and higher [8]. This opened up a path to the study of approximate versions of the Hanani-Tutte theorem [9, 11].

We complement these results by proving that the Hanani-Tutte theorem does extend to the torus.

▶ **Theorem 1.** *Let G be a graph. Suppose that G can be drawn on the torus so that every two independent edges cross evenly. Then G can be embedded on the torus.*

Among orientable surfaces, this leaves only the double and triple torus for which we do not yet know whether the Hanani-Tutte theorem holds. For non-orientable surfaces, all cases starting with the Klein bottle are open. Our approach extends and refines techniques developed in [4, 25] and other papers.

[1] Crossings for us are proper crossings, not shared endpoints or touching points.
[2] Equivalently, a sphere with a crosscap. We assume that the reader is familiar with the basic terminology of drawings and embeddings in surfaces. For background see [6, 21].




The proof of Theorem 1 is inductive, and for the induction to work we need to strengthen the result; this strengthened version is Theorem 4 in Section 3. For the induction, we carefully define a partial order on drawings of graphs on the torus, then consider a minimal counterexample to Theorem 4 with respect to the partial order. Our reductions are mostly, but not always, minor preserving.

In the base case of the proof, we work with drawings of subdivisions of $K_{3,t}$, $t \leq 6$ and K_5, possibly with some added paths in the case of $K_{3,t}$. Reducing to the base case is a relatively smooth procedure using natural redrawing tools, up to the point when the graph is a subdivision of $K_{3,t}$ or K_5 with additional simple "bridges". In this case an extensive case analysis seems to us the only way to proceed. The bulk of the full version of the paper deals with these cases. The main difficulty then lies in performing the reduction steps so that the case analysis is manageable.

Applications

Smorodinsky and Sharir [33] showed that if P is a collection of n points, and C a collection of m pseudo-disks in the plane such that every pseudo-disk in C passes through a distinct pair of points in P, and no pseudo-disk contains a point of P in its interior, then $m \leq 3n-6$, where $3n-6$ is just the maximal number of edges in a planar graph on n vertices. For pseudodisks in the projective plane, we have $m \leq 3n-3$ [32]. Using Theorem 1, one can now obtain $m \leq 3n$ for pseudodisks in the torus.

A family of simply connected regions is *k-admissible* if each pair of region boundaries intersect in an even number of points, not exceeding k. Whitesides and Zhao [36] showed that the union of $n \geq 3$ planar k-admissible sets is bounded by at most $k(3n-6)$ arcs, where $3n-6$ is again the maximum number of edges in an n-vertex planar graph. This result was reproved by Pach and Sharir [23] using the Hanani-Tutte theorem. Consequently, we can get a bound of $k(3n-3)$ for the projective plane [32] and $3kn$ for the torus.

Keszegh [17] gave a more uniform treatment of these types of results based on hypergraphs, which at the core again uses the Hanani-Tutte theorem. While he only states results for the plane, all of his material should lift to the projective plane and the torus based on the availability of the Hanani-Tutte theorem on those surfaces.

Known Results

The Hanani-Tutte theorem [2, 35] for the plane has been known for a while, with many proofs; for a survey of known results see [32]. It was shown to be true for the projective plane by Pelsmajer et al. [25] using the excluded minors for the projective plane, and later directly, without recourse to excluded minors, by É. Colin de Verdière et al. [4]. Excluded minors stop being useful at this point, since we do not know the complete list of excluded minors for the torus or any higher-order surfaces.

If we strengthen the assumption of the Hanani-Tutte theorem to also require adjacent edges to cross each other evenly, then embeddability follows, for any surfaces. This is known as the weak Hanani-Tutte theorem.[3]

▶ **Theorem 2** (Weak Hanani-Tutte for Surfaces [1, 28]). *If a graph can be drawn in a surface S (orientable or not) so that every two edges cross an even number of times, then the graph can be embedded in S with the same rotation system (if S is orientable), or the same embedding scheme (if S is non-orientable).*

[3] Naming this variant "weak" is somewhat misleading, as it has a strengthened conclusion.

Unfortunately, the available proofs of Theorem 2 do not give any insight on how to establish the strong version on a surface (when it is possible). It does not even settle the seemingly easy question of whether a graph which can be drawn in a surface so that the only crossings are between adjacent edges, can always be embedded in that surface.

For work on applying Hanani-Tutte to different planarity variants, see [5, 12, 13, 14, 16, 31]. A variant of the (strong) Hanani-Tutte theorem in the context of approximating maps of graphs, which works on any surface, was announced in a paper co-authored by the first author [7].

Organization

We introduce the terminology and basic redrawing tools in Section 2 and 3, respectively. As a sample applicaion, in Section 4 we extend the Hanani-Tutte theorem to the 1-spindle (a pseudosurface). In Section 5, we discuss the base case of our inductive proof of Theorem 1. In Section 6, we discuss the properties of a hypothetical minimal counterexample to Theorem 1. In Section 7 we outline the proof of Theorem 1. We conclude with open problems in Section 8.

2 Terminology, Definitions, and Basic Properties

For the purposes of this paper, graphs are simple (no multiple edges or loops), and surfaces are compact 2-manifolds (with or without boundary).

For a particular drawing D of a graph G on a surface S, we make the following definitions: The *crossing parity* of a pair of edges is the number of times the two edges cross modulo 2. Two edges form an *odd pair* if their crossing parity is 1. A subgraph (or single edge) is *even* if none of its edges belong to an odd pair in G.

A closed curve γ on a surface S is *non-essential* if it forms the boundary of a (not necessarily connected) closed sub-manifold of S, or equivalently, if its complement in S can be two-colored so that path-connected components sharing a non-trivial part of γ receive opposite colors. In homological terms, a closed curve γ on S is *essential* if its homology class does not vanish over $\mathbb{Z}_2$. A closed curve *separates the surface* if removing the curve from the surface disconnects the surface. A curve is *simple* if it is free of self-intersections. A closed simple curve γ separates the surface if and only if γ is non-essential. An essential simple closed curve is therefore *non-separating*.

We apply the same terminology to cycles in graph drawings, since a cycle determines a closed curve. We say a subgraph in a drawing is *essential* if it contains an essential cycle.

Closed curves in the plane are non-essential. Non-essential curves, on any surface, tend to be easier to handle when proving results similar to ours, because they cross every other closed curve an even number of times. (This is easy to see by two-coloring of the complement of the non-essential curve as described above.) The greater difficulty in proving Hanani-Tutte type results lies in the presence of essential curves.

Edge-vertex move

An *edge-vertex* (e, v)*-move* (also known as *van Kampen's finger-move*, or *edge-vertex switch*) is a generic deformation of the edge e in a drawing of G changing the crossing parity between e and all the edges incident to v, without changing any other crossing parities; see the left illustration in Figure 1. An edge-vertex move is performed as follows. Connect an interior point of e to v via a curve γ that does not pass through any vertices (and does not cross e). Then reroute e close to γ and around v (as shown in the illustration). Since e traverses γ twice, only the crossing parities of e with edges incident to v change.

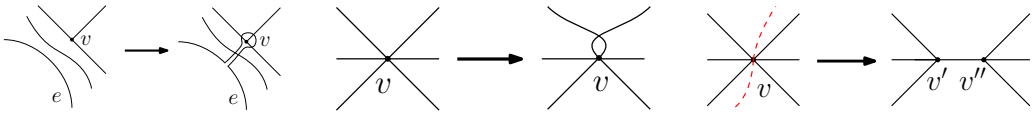


Figure 1 *Left:* An (e, v)-move. *Middle:* An edge-flip. *Right:* A vertex-split.

Edge-flip

An *edge-flip*, or *flip*, *(at v)* in a drawing is a redrawing operation that happens near a vertex v, and which takes two consecutive edges in the rotation at v and (locally) exchanges their position in the rotation at v; see the middle illustration in Figure 1. As a result, the crossing parity between the two flipped edges changes, and no other crossing parities are affected.

Edge contraction and vertex split

A *contraction* of an edge $e = uv$ in a drawing of a graph is an operation that turns e into a vertex by moving v along e towards u while dragging all the other edges incident to v along e. If e is even, then contracting e in this fashion does not change the crossing parity of any pair of edges. Contraction may introduce multi-edges or loops at the vertices; we avoid this by only contracting partially, just enough to make uv crossing-free.

We will also often use the following operation which can be thought of as the inverse of a contraction in a drawing. To *split* a vertex v, we split its rotation into two contiguous parts, and then cut through the vertex to separate those two parts. This results in two vertices v' and v'' which we connect by a crossing-free edge $v'v''$ so that contracting $v'v''$ recovers the original rotation at v; see the right illustration in Figure 1. Vertex splits are not unique.

Edge-vertex moves and contractions may introduce self-crossings of edges. Such self-crossings are easily resolved [15, Section 3.1]: remove the crossing, and reconnect the four severed ends so that the edge consists of a single curve. In this redrawing, essential cycles remain essential and non-essential cycles remain non-essential. (This is easy to see by two-coloring of the complement of the curve corresponding to a non-essential cycle.)

3 Redrawing iocr-0-Drawings on the Torus

In this section we establish some of the basic redrawing tools for iocr-0-drawings. We start with some results which (mostly) work on all surfaces, and may be useful for establishing Hanani-Tutte type results for surfaces other than the torus. We focus on orientable surfaces.

We will say that a vertex v in a drawing D of a graph on a surface is *even* if every two edges incident to v cross each other an even number of times; otherwise, v is *odd*.[4]

A drawing D_2 of a graph G in an orientable surface S is *compatible* with a drawing D_1 of G in S if every even vertex in D_1 is even in D_2 and the rotation at each even vertex in D_1 is preserved in D_2. Note that the compatibility relation is transitive, but not necessarily symmetric. We define a notion of connectivity on even vertices: two even vertices u and v in a drawing of a graph are *evenly connected* if there exists a path connecting u and v consisting only of even vertices. An *evenly connected component* in a drawing is a maximal connected subgraph in the underlying abstract graph induced by a set of even vertices.

[4] This will not conflict with the usual degree-based definition of even and odd vertices, since we will not be using that terminology.

A drawing D_2 of a graph G in an orientable surface S is *weakly compatible* with a drawing D_1 of G in S if every even vertex in D_1 is even in D_2 and, for every evenly connected component K in D_1, either the rotation in D_2 at every vertex $v \in V(K)$ is the same as the rotation of v in D_1, or the rotation in D_2 at every vertex $v \in V(K)$ is the reverse of the rotation of v in D_1. Note that the weak compatibility relation is transitive, but not necessarily symmetric, just like compatibility.

We can now state a version of the Hanani-Tutte theorem on the plane that implies both the weak and the strong version. While the result follows from the proof of the Hanani-Tutte theorem in [26], it was first explicitly stated, and given a new proof, in [10].

▶ **Theorem 3** (The Unified Hanani-Tutte Theorem [10])**.** *If G has an iocr-0-drawing D in the plane, then G has an embedding in the plane compatible with D.*

Theorem 3 does not hold on any surface other than the plane [8, Theorem 7]. The counterexample requires compatibility; it fails for weak compatibility. If we assume that the graph is 3-connected, then weak compatibility can be achieved on the torus.

▶ **Theorem 4.** *If a 3-connected graph G has an iocr-0-drawing D in the torus, then G has an embedding in the torus that is weakly compatible with D.*

The next lemma is one of our main redrawing tools. It shows that we can always clear an essential cycle of crossings, in any surface. A precursor of this lemma, for the projective plane can be found in [25].

▶ **Lemma 5.** *Let D be an iocr-0-drawing of a graph G on a surface S. Suppose that C is an essential cycle in D. Then there exists an iocr-0-drawing D' of G compatible with D on S in which C is crossing free, and each cycle is essential in D' if and only if it is essential in D.*

As a first application of Lemma 5 we obtain the following corollary.

▶ **Corollary 6.** *If G has an iocr-0-drawing D on the torus containing an essential cycle C consisting of even vertices only, then G has an embedding on the torus that is compatible with D.*

The following lemma shows, roughly speaking, that in an iocr-0-drawing, one vertex alone cannot make the difference between planarity and non-planarity. (It appeared, with a slightly different proof, for the projective plane in [25].)

▶ **Lemma 7.** *Let $x \in V(G)$ and $H = G - x$. Suppose there is an iocr-0-drawing D of G in a surface S such that H is non-essential. Then G has a plane embedding. If S is orientable, then the plane embedding of G restricted to H is compatible with D restricted to H.*

Lemma 7 implies that if a graph H has a non-essential embedding on a surface, then H has a compatible plane embedding.

Proof. We consider the surface S as a sphere with handles and crosscaps. We can choose a set $\mathcal{C}$ of 1-sided and 2-sided essential curves so that cutting the surface along these curves in $\mathcal{C}$ results in a planar surface, with a single hole for each crosscap and handle[5]: for each crosscap we choose a 1-sided closed curve cutting through the crosscap, and for each handle we pick two 2-sided closed curves (sharing a single point) so that cutting the surface along the two curves results in a single "square" boundary hole.

[5] For non-orientable surfaces Mohar [20] calls these "planarizing system of disjoint curves". If we deformed the curves of $\mathcal{C}$ so that they all shared a single point, we'd get a set of generators of the fundamental group of the surface.

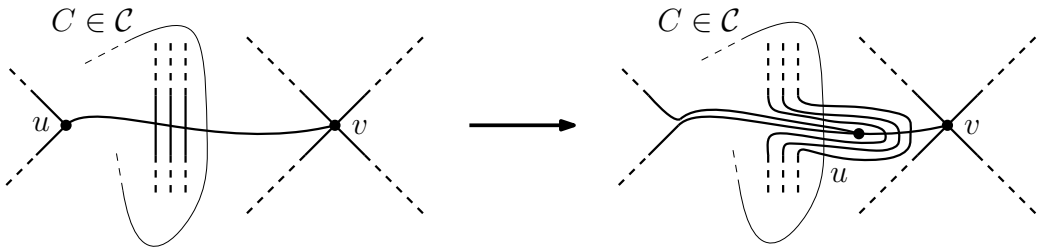


Figure 2 Contracting the edge uv by pulling u along uv to v.

Given an edge uv, we may contract the edge by pulling u toward v until uv no longer crosses any curve of $\mathcal{C}$. For any edge e that crosses uv, when u reaches e then e will be deformed so that instead of u crossing e, e will get pulled along to just stay in front of u, see Figure 2. This may cause e to add new crossings with curves of $\mathcal{C}$, two crossings at a time, whenever u crosses a curve of $\mathcal{C}$. Thus the crossing parity between each edge of G not incident to u and each curve of $\mathcal{C}$ will be unchanged by this operation.

Let F be a rooted maximum spanning forest in $H := G - x$. For each component of F, perform a breadth-first search transversal, contracting each edge as described towards the root of the component. After an edge e of F is contracted it has zero crossings with every curve of C; later contractions may add crossings between e and curves of $\mathcal{C}$ but without affecting the crossing parity. Thus, at the end of this process, every edge of F crosses every curve of $\mathcal{C}$ evenly.

Consider any edge $e \in E(H) - E(F)$. Since we assumed that H is non-essential (and that did not change during the redrawing we did), e must cross each $C \in \mathcal{C}$ evenly (if it did not, the (unique) cycle in $F \cup \{e\}$ is essential, a contradiction). In summary, every edge of H crosses each $C \in \mathcal{C}$ evenly.

Now cut the surface along the curves in $\mathcal{C}$, and fill in each boundary hole with a disk, resulting in a sphere. For each edge-crosscap intersection, reconnect the ends with a curve passing straight through the disk; for each edge broken at a handle, reconnect it straight across the disk. We remove any resulting self-crossings using the standard method [15, Section 3.1]. Since each edge of H intersects each curve of $\mathcal{C}$ evenly, its crossing parity with other edges does not change. In particular: if two edges did not cross oddly before the redrawing, but they do now, then both edges are incident to x, and even vertices, except possibly x, remain even.

We can then apply Theorem 3 to obtain a plane drawing of G. If S is orientable, then the subdrawing of H is compatible to the original drawing of H on S. ◀

The following result shows that an even tree (all its edges are even) can always be cleaned of crossings. The result is true for all surfaces, and it remains true for forests, but we will not need these stronger versions.

▶ **Lemma 8.** *If G has an iocr-0-drawing D that contains an even tree T, then there exists an iocr-0-drawing of G that is compatible with D in which T is free of crossings. Only edges incident to T are redrawn.*

A pair of edge-disjoint cycles C_1 and C_2 *touch* at a vertex $v \in V(C_1) \cap V(C_2)$ if in the rotation at v the edges of C_1 and C_2 do not interleave; otherwise, we say the cycles *cross* at v. A pair of *nearly disjoint* cycles is a pair of edge-disjoint cycles for which any shared vertices are even and touching.

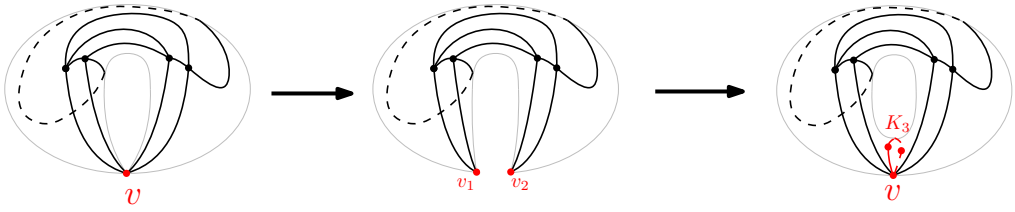


Figure 3 Surgery in the proof of Theorem 10. Eliminating the pinched point splits the vertex v into two vertices v_1 and v_2 that are subsequently merged using the handle. A 3-cycle K_3 is added around the handle passing through v.

▶ **Lemma 9.** *If G has an iocr-0-drawing on the torus containing two nearly disjoint essential cycles, then G has an embedding on the torus that is compatible with the iocr-0-drawing, and the two nearly disjoint cycles remain essential.*

This lemma implies that we can assume that a counterexample to the Hanani-Tutte theorem on the torus does not contain two vertex-disjoint essential cycles.

4 The 1-Spindle

Before we move on to the torus, we show how to apply our tools so far in a much simpler context than the torus (or the projective plane).

We show that there is a strong Hanani-Tutte theorem for the 1-spindle. The 1-*spindle* is a pseudosurface obtained from the sphere by identifying two distinct points (the *pinchpoint*). Embeddings are defined as usual; for drawings, we allow at most one edge to pass through the pinchpoint.

▶ **Theorem 10** (Hanani-Tutte for 1-Spindle)**.** *If a graph has an iocr-0-drawing on the 1-spindle, it can be embedded on the 1-spindle. The embedding is compatible with the original drawing, except for (possibly) the vertex at the pinchpoint.*

One might suppose that this result should follow directly from a Hanani-Tutte theorem for the projective plane or the torus, but if true, it does not seem to be immediately obvious.

Proof. Fix an iocr-0-drawing of G on the 1-spindle. We can assume that there is a vertex v at the pinchpoint. If not, there must be an edge e passing through the pinchpoint (otherwise, we have an iocr-0-drawing on the sphere, and we are done by the Hanani–Tutte theorem in the plane). Consider the subcurve γ of e between the pinchpoint and a vertex v. For any edge f which crosses γ oddly, we perform an (f, u)-move for every vertex u in G. This does not change the crossing parity of any pair of edges, but it does ensure that γ is an even curve. We can then partially contract e by pulling v along γ onto the pinchpoint. Since γ is even, the drawing remains iocr-0.

Refer to Figure 3. Remove the pinchpoint (splitting v into two copies), giving us a drawing on a sphere. Add a handle close to the two copies (close to where they were split), and move them back together, merging them into a single vertex. This gives us an iocr-0-drawing of G on the torus. Add a crossing-free essential cycle, say a 3-cycle K_3, through v on the handle; let G^* denote the new graph. If the drawing of $G - v$ (as part of G^*) does not contain an essential cycle, then G is planar, by Lemma 7, and we are done. So the drawing of $G - v$ contains an essential cycle, implying that the drawing of G^* contains two disjoint essential cycles. By Lemma 9 we can find a compatible embedding of G^* in the torus. Note that in that embedding the essential cycle K_3 is still essential (and free of crossings), so we can contract it until it becomes a point, which yields an embedding of G on the 1-spindle. ◀

5 Hanani-Tutte for Some Kuratowski Minors

Fulek and Kynčl [11] showed that the Hanani-Tutte theorem is true for any surface if we restrict ourselves to the graphs known as Kuratowski minors, which include $K_{3,t}$ for any $t \geq 3$.

▶ **Lemma 11** ([11]). *For $t \geq 7$, $K_{3,t}$ does not admit an iocr-0-drawing on the torus.*

Since $K_{3,7}$ has no toroidal embedding, Lemma 11 implies that the Hanani-Tutte theorem on the torus is true for all $K_{3,t}$, $t \geq 7$ (and their subdivisions). Figure 4 shows a toroidal embedding of $K_{3,6}$ with an 3-cycle added to the vertices of degree 6, so Lemma 11 is sharp even if with that added 3-cycle.

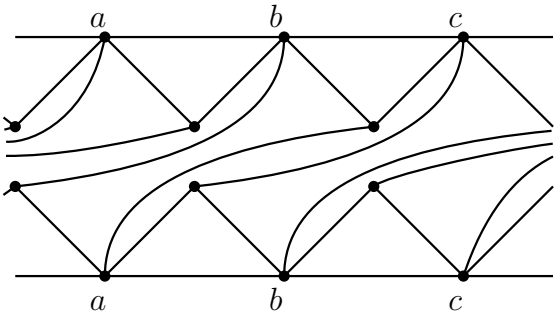


Figure 4 A torus embedding of $K_{3,6} \cup K_3$, with the union taken over the three vertices of degree 6.

As a base case for our proof of Theorem 1, we need a unified Hanani-Tutte result like Theorem 3 for Kuratowski minors on the torus. Fulek and Kynčl [8] showed that this is not possible, even for a $K_{3,4}$: there is an iocr-0-drawing of $K_{3,4}$ on the torus which does not have a compatible embedding. We can show, however, that there always is a *weakly* compatible embedding of $K_{3,n}$ up to $n = 6$ which is sharp, since $K_{3,7}$ has no embedding on the torus.

We establish a slightly stronger result. Let a, b, and c be the three vertices of degree n in $K_{3,n}$. We call a graph a $K_{3,n}$ *with bracers (at a, b, c)* if it is the union of the $K_{3,n}$ with (any number of) internally-disjoint paths of length at most two added between any two of $\{a, b, c\}$ (no multiple edges). These paths are the *bracers*.

▶ **Lemma 12.** *Let G be a subgraph of a subdivision of a $K_{3,n}$ with bracers at $\{a, b, c\}$. For every iocr-0-drawing of G on the torus, there exists a weakly compatible embedding of G.*

The result remains true for bracers of arbitrary length, but we will not need that.

Proof. We can assume that G contains no vertex v of degree 0 or 1, since removing such a vertex cannot make another even vertex odd, and any embedding of $G - v$ can easily be extended to G. If v is a vertex of degree 2 with neighbors u, w with $uw \notin E(G)$, then we can suppress v by similar reasoning. Thus we may assume that G is an induced subgraph of a $K_{3,n}$ with bracers.

Let $S = \{a, b, c\}$. We want to replace bracers of length 1 with bracers of length 2. So suppose there is an edge uw with $u, w \in S$. If necessary, we use edge-flips at u to ensure that uv crosses every edge incident to u evenly. We can then subdivide uw with a vertex v placed very close to u; then even vertices stay even, and after embedding we can suppress v. Thus we may assume that all bracers are paths of length 2. Finally, we may assume that all the vertices of degree at most 3 are even, by performing edge-flips if necessary. So any odd vertices must belong to S.

If S contains no even vertices, then each even vertex of G is its own evenly connected component. Since any two rotations of a degree-3 vertex are weakly compatible, any embedding of G on the torus suffices, which we have from Figure 4. If S contains only even vertices, we are done by Theorem 2.

Suppose that S contains exactly one odd vertex. If there is an essential cycle that avoids the odd vertex in S, we are done by Corollary 6. Otherwise, all essential cycles pass through the odd vertex in S, in which case we are done by Lemma 7.

This leaves us with the case that S contains two odd vertices and one even vertex. This case is more complicated and its proof can be found in the full version of the paper. ◀

We also need a unified Hanani-Tutte theorem for K_5.

▶ **Lemma 13.** *For every iocr-0-drawing D of a subgraph G of a subdivision of K_5 on the torus, there exists an embedding of G on the torus that is compatible with D.*

6 Minimal Counterexamples

In this section we collect properties of a minimal counterexample to the Hanani-Tutte theorem on the torus. We order graphs first by their number of isolated vertices (**in**creasing), then by the number of edges (**in**creasing), and finally by the number of vertices (**de**creasing). We denote by $\prec$ the strict partial ordering based on these three numbers, and refer to a $\prec$-minimal graph (with certain properties). We write minimal, rather than $\prec$-minimal.

Since isolated vertices do not affect embeddability on a surface, nor the Hanani-Tutte criterion, a minimal counterexample contains no isolated vertices. Graphs without isolated vertices are then ordered by number of edges (increasing), and number of vertices (decreasing); equivalently, we use the (strict, partial) lexicographic order on $(|E(G)|, 2|E(G)| - |V(G)|)$; since graphs without isolated vertices satisfy $|V(G)| \leq 2|E(G)|$, this order is well-founded. Note that if H is a proper subgraph of G, then $H \prec G$, simply because it has fewer edges (since G has no isolated vertices).

Section 6.1 presents some basic properties of minimal counterexamples with respect to cycles and cuts. In Section 6.2 we identify what we call an X-configuration, which must occur in a minimal counterexample to Theorem 1, the Hanani-Tutte theorem on the torus. One issue we will not further discuss in this extended abstract, is that the proof of Theorem 1 requires a strengthened assumption on (weakly) compatible embeddings, for which we do not know how to show that an X-configuration occurs.

6.1 Nearly Disjoint Cycles and Cuts

The next lemmas are true whether "weakly" is included or omitted.[6]

▶ **Lemma 14.** *If G is a minimal graph that has an iocr-0-drawing D on the torus, but* does not *have a [weakly] compatible embedding on the torus, then D does not contain a pair of nearly-disjoint cycles at least one of which is essential.*

We use the previous lemma to repeatedly reduce a minimal counterexample G to the Hanani-Tutte theorem on the torus, to show that it does not exist.

[6] It is tempting to assume that the weakly compatible version of the lemma implies the compatible version, but this is not necessarily the case, since the minimal counterexample in the two cases may be different, and therefore have different properties.

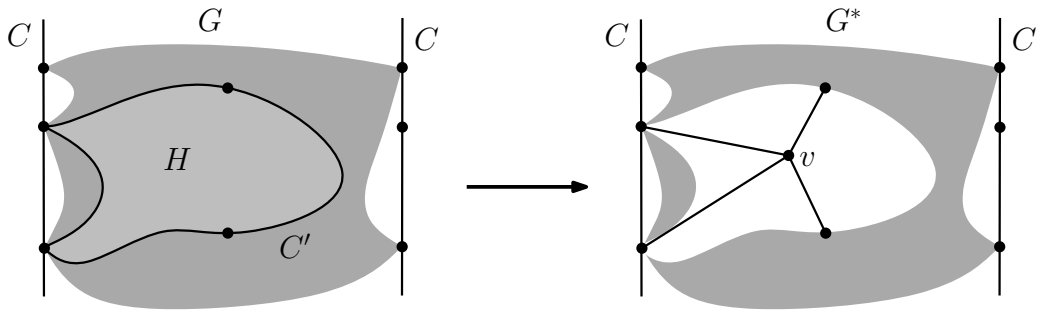


Figure 5 Construction of a reduced graph G^* drawn on a cylinder, illustrating a case in the omitted proof of Lemma 15. C and C' are weakly disjoint cycles. C is essential so we are able to clear it of crossings, then cut along it, creating a cylinder bounded by two copies of C. Replacing the subgraph H interior to C' with a vertex v constructs G^* from G. Note that $G^* \prec G$.

The following lemma allows us to focus on 2-connected (for compatibility and weak compatibility) or 3-connected counterexamples (in general).

▶ **Lemma 15.** *If G is a minimal graph D that has an iocr-0-drawing D on the torus, but does* not *have a [weakly] compatible embedding on the torus, then G is 2-connected and it has no 2-cut consists of two odd vertices of D.*

If we do not require the embedding to be [weakly] compatible, then a minimal counterexample must be 3-connected.

6.2 The X-Configuration

The Hanani-Tutte theorem holds for cubic graphs on arbitrary surfaces; this is because vertices of degree 3 can be made even by edge-flips, at which point Theorem 2 can be applied.

Consider a vertex v incident to four edges e_1, e_2, e_3, e_4 which occur in (cyclic) order e_1, e_2, e_3, e_4 in the rotation at v. Suppose that e_1 and e_3 cross oddly, and every other pair of (these four) edges crosses evenly. No matter what edge-flips are performed at v, there will always remain at least one pair of edges crossing oddly. Moreover, this configuration is the unique obstacle to a vertex being even, up to edge-flips: suppose v is incident to four edges. Using edge-flips we can make three consecutive edges at v cross evenly with each other. Say the edges are e_1, e_2, e_3, e_4, in this order, and every two of e_1, e_2 and e_3 cross evenly. If e_4 crosses exactly e_2 oddly, we are done. If it crosses e_3 and e_1 oddly, we move the end of e_4 once around v, so that e_4 crosses exactly e_2 oddly. In all other cases, e_4 and the edges that it crosses oddly form a consecutive subset of the cyclic order, so e_4 can be made to cross all of e_1, e_2 and e_3 evenly using edge-flips.

Hence, the edges incident to a vertex v of degree 4 are equivalent via edge flips to one of two configurations, either with no odd crossings or with a single odd crossing between a pair of non-consecutive edges in the rotation at v. The next lemma shows that four edges are always the obstacle to making a vertex even by flips, independent of its degree.

▶ **Lemma 16.** *If a vertex in a drawing cannot be made even by flips, then it is incident to four edges which cannot be made to cross each other evenly by flips.*

Lemma 16 can also be found in the full version of [7, Claim 8].

The core of the inductive proof will be working with a specific configuration in iocr-0-drawings: Two essential cycles C_1 and C_2 which intersect in a single vertex v so that the edges of C_1 and C_2 incident to v cannot be made to cross each other evenly by edge-flips at v. We call such a pair (C_1, C_2) an *X-configuration*, see Figure 6.

Figure 6 An X-configuration (left). At illustration of the construction of an X-configuration (C_1, C_2) in the proof of Lemma 17; crossings are not shown.

▶ **Lemma 17.** *If G is a minimal 3-connected graph that has an iocr-0-drawing D on the torus, but* does not *have an embedding on the torus, then D contains an X-configuration (C_1, C_2).*

We emphasize that Lemma 17 does not require the embedding to be compatible, or weakly compatible.

Proof. If every vertex of G in an iocr-0-drawing of G on the torus can be made even by flips, we can obtain an embedding of G by Theorem 2, the weak version of the Hanani-Tutte theorem. Therefore, there exists a vertex v for which this is not the case. By Lemma 16 there are four edges $e_i = vu_i$, $1 \le i \le 4$ incident to v which cannot all be made to cross each other evenly by edge-flips.

Refer to Figure 6 (on the right). Using the 3-connectedness of G, we will find edge-disjoint cycles C_1, C_2 intersecting in v as follows: Consider a minimal path in G connecting two u_i vertices that avoid v; without loss of generality this path is P_{12} between u_1 and u_2 in $G - \{v, u_3, u_4\}$. Let C_{12} denote the cycle obtained as the edge union of $E(P_{12}), \{e_1\}$ and $\{e_2\}$. Let P_3 denote a path in $G - \{v, u_4\}$ between u_3 and $V(C_{12}) - \{v\}$. Let w denote the end vertex of P_3 on $C_{12} - \{v\}$. Finally, let P_4 denote a path in $G - \{v, w\}$ between u_4 and $(V(P_3) \cup V(C_{12})) - \{v, w\}$. No matter where P_4 ends, the edge union of $E(C_{12}), E(P_3), E(P_4), \{e_3\}$ and $\{e_4\}$ contains a pair of cycles C_1 and C_2 meeting exactly in v.

As explained before Lemma 16, using edge-flips we can assume that the e_i cross each other evenly, with the exception of one pair of edges which are not consecutive at v. If the ends of C_1 and C_2 do not alternate at v, then the two edges crossing oddly cannot both belong to the same C_i, since they would then be consecutive; therefore one belongs to C_1 and the other to C_2, which implies that C_1 and C_2 cross each other oddly (all edges of the cycles not incident to v cross evenly, and the two cycles do not cross at v), hence both cycles are essential. If the ends of C_1 and C_2 do alternate at v, then the two edges crossing oddly must belong to the same cycle (since otherwise they would be consecutive), so again C_1 and C_2 cross each other oddly (at v, no other odd crossings), and both are essential. ◀

7 Proof of Theorem 1

Assume, for a contradiction, that Theorem 1 fails. Then there is a graph G with an iocr-0-drawing D on the torus, such that G cannot be embedded on the torus. Let G be a minimal such counterexample, with iocr-0-drawing D. By Lemma 15 we know that G is 3-connected, and by Lemma 17 that D contains an X-configuration (C, C_0). So there is a counterexample to the following statement, a strengthened version of Theorem 1:

> Let D be an iocr-0-drawing of a graph G containing an X-configuration (C, C_0), then G has a weakly compatible embedding on the torus.

If the statement is true, we say that G – or more precisely, (G, D, C, C_0) – satisfies *HT-XWC*, acronym for Hanani-Tutte–X-Weakly-Compatibility. So we know that there is a (G, D, C, C_0) which does not satisfy HT-XWC, and therefore a minimal such counterexample.

The omitted technical parts of the proof proceed by reducing the 4-tuple (G, D, C, C_0) using Lemma 14 and similar technical tools. In the end, an application of Lemma 12 or Lemma 13 shows that no counterexample violating HT-XWC exists, which is the desired contradiction.

8 Open Questions

8.1 Crossing Elimination

Theorem 1 implies that if a graph can be drawn on the torus so that its only crossings are between adjacent edges, then it is toroidal. This might appear to be intuitively clear, but for the plane, for the projective plane, and now for the torus, we only know it to be true by virtue of their respective Hanani-Tutte theorems; even for the plane no simpler proof is known. Since we know that the Hanani-Tutte theorem does not hold for orientable surfaces of genus at least 4 [8], this begs the question whether we can always remove adjacent crossings.

▶ **Conjecture 18.** *If a graph can be drawn in a surface without any independent crossings, then the graph is embeddable in that surface.*

The counterexample from [8] requires independent edges to cross (evenly), so it does not resolve this conjecture.

The Hanani-Tutte theorem is related to the *independent odd crossing number* $\operatorname{iocr}(G)$ of a graph G, the fewest number of pairs of independent edges that have to cross oddly in a drawing of G (see [24, Section 6]). The Hanani-Tutte theorem then states that $\operatorname{iocr}(G) = 0$ implies that $\operatorname{cr}(G) = 0$, where $\operatorname{cr}(G)$ is the traditional crossing number of G. It is even true that $\operatorname{iocr}(G) = \operatorname{cr}(G)$ for $\operatorname{iocr}(G) \leq 2$ [29], though we know that there are graphs G for which $\operatorname{iocr}(G) < \operatorname{cr}(G)$ [27]. The two crossing numbers cannot be arbitrarily far apart though, one can show that $\operatorname{cr}(G) \leq \binom{2\operatorname{iocr}(G)}{2}$ [29]. It is not known whether similar bounds, or any bounds, for that matter, hold for other surfaces, not even the projective plane. (The subscript in crossing number variants indicates the surface we work on.)

▶ **Conjecture 19.** $\operatorname{cr}_\Sigma(G) \leq \binom{2\operatorname{iocr}_\Sigma(G)}{2}$, *where* Σ *is the projective plane, or the torus.*

We also now have a crossing lemma for iocr on the torus.

▶ **Corollary 20** (Crossing Lemma). $\operatorname{iocr}_T(G) \geq cm^3/n^2$, *where* T *is the torus,* $c = 1/64$, $m = |E(G)|$, *and* $n = |V(G)|$.

The proof follows from the standard crossing lemma argument done carefully combined with Theorem 1 (see the section on crossing lemma variants in [30]). Since iocr lower bounds several other crossing number variants, such as iacr, pcr, and pcr_ this also implies a crossing lemma for these variants. Also see [18] for a sketch of an argument that shows a crossing lemma for iocr for arbitrary surfaces, but without explicit constants c.

The proof of Theorem 1 is algorithmic in the sense that given an iocr-0-drawing of G we can find an embedding of G in the torus (all reductions in the lemmas are constructive in that sense, so the minimal counterexample assumption can be turned into a recursion). This does not imply that testing whether a graph can be embedded in the torus lies in polynomial time. The planar Hanani-Tutte theorem can be turned into a linear system of equations over $\mathbb{Z}_2$, which is solvable in polynomial time (see [32, Section 1.4.2]). Unfortunately, modeling the

handle of the torus requires quadratic equations, which loses us polynomial-time solvability. This is similar to the situation for the projective plane, where the projective handle also leads to quadratic equations.

▶ **Question 21.** *Can the Hanani-Tutte criteria for the projective plane and the torus be turned into a polynomial-time test?*

Such tests would not be competitive with existing algorithms – the running time in the plane is $O(n^6)$, but they have the potential to be significantly simpler (as is true for the plane).

8.2 Arf and Approximating the Hanani-Tutte Theorem

The fact that the Hanani–Tutte theorem cannot be extended to all orientable surfaces has some positive practical consequences. Some results about graphs embedded on a surface remain true if we only require that the graph has an independently even drawing on the surface, and this is a strictly larger class of graphs in general (starting at genus 4).

One notable example is the Arf invariant formula for the number of perfect matchings in a graph embedded on an orientable surface [19, Remark 1.4, Theorem 1][7], see also [3, 22]. The proof of the similar result by Tesler [34], which applies to all closed surfaces, still works for independently even drawings instead of embeddings. In both cases, the complexity of the formula depends exponentially on the genus of the underlying surface, so for graphs which have an independently even drawing in a surface in which they cannot be embedded, the complexity of the formula is improved; its length does not depend on the genus of the graph, but rather its $\mathbb{Z}_2$*-genus*, the smallest genus of a surface on which the graph has an independently even drawing.

It is therefore interesting to study how large the gap between the genus and the $\mathbb{Z}_2$-genus of a graph may be. It is known that the gap can be at least linear [8, Corollary 4.1]. There also is an upper bound on this gap, but it is not effective [11].

References

1 Grant Cairns and Yury Nikolayevsky. Bounds for generalized thrackles. *Discrete Comput. Geom.*, 23(2):191–206, 2000.

2 Chaim Chojnacki (Haim Hanani). Über wesentlich unplättbare Kurven im drei-dimensionalen Raume. *Fundamenta Mathematicae*, 23:135–142, 1934.

3 David Cimasoni and Nicolai Reshetikhin. Dimers on surface graphs and spin structures. i. *Communications in Mathematical Physics*, 275(1):187–208, 2007.

4 Éric Colin de Verdière, Vojtěch Kaluža, Pavel Paták, Zuzana Patáková, and Martin Tancer. A direct proof of the strong Hanani-Tutte theorem on the projective plane. *Journal of Graph Algorithms and Applications*, 21(5):939–981, 2017. doi:10.7155/jgaa.00445.

5 Hooman R. Dehkordi and Graham Farr. Non-separating planar graphs. *Electron. J. Comb.*, 28(1):P1.43, 16, 2021.

6 Reinhard Diestel. *Graph theory*, volume 173 of *Graduate Texts in Mathematics*. Springer-Verlag, Berlin, third edition, 2005.

[7] The authors attribute the extension of the result to independently even drawings to Norine. The extension is even more general, since instead of independently even drawings Norine considers so-called (perfect) matching even drawings, in which every perfect matching induces an even number of crossings. It is easy to see that independently even drawings form a proper sub-class of matching even drawings on every surface.

7 Radoslav Fulek and Jan Kynčl. Hanani–Tutte for approximating maps of graphs. In *34th International Symposium on Computational Geometry, SoCG 2018, June 11-14, 2018, Budapest, Hungary*, pages 39:1–39:15, 2018. (full version: arXiv:1705.05243. doi:10.4230/LIPIcs.SoCG.2018.39.

8 Radoslav Fulek and Jan Kynčl. Counterexample to an extension of the Hanani-Tutte theorem on the surface of genus 4. *Combinatorica*, 39(6):1267–1279, 2019.

9 Radoslav Fulek and Jan Kynčl. $\mathbb{Z}_2$-Genus of Graphs and Minimum Rank of Partial Symmetric Matrices. In *35th International Symposium on Computational Geometry (SoCG 2019)*, Leibniz International Proceedings in Informatics (LIPIcs), pages 39:1–39:16, 2019.

10 Radoslav Fulek, Jan Kynčl, and Dömötör Pálvölgyi. Unified Hanani-Tutte theorem. *Electr. J. Comb.*, 24(3):P3.18, 2017.

11 Radoslav Fulek and Jan Kynčl. The $\mathbb{Z}_2$-genus of Kuratowski minors. In *34th International Symposium on Computational Geometry, SoCG 2018, June 11-14, 2018, Budapest, Hungary*, pages 40:1–40:14, 2018. doi:10.4230/LIPIcs.SoCG.2018.40.

12 Radoslav Fulek, Michael Pelsmajer, Marcus Schaefer, and Daniel Štefankovič. Hanani-Tutte, monotone drawings, and level-planarity. In János Pach, editor, *Thirty Essays on Geometric Graph Theory*, pages 263–287. Springer, 2013.

13 Radoslav Fulek, Michael J. Pelsmajer, and Marcus Schaefer. Hanani-Tutte for radial planarity II. In Yifan Hu and Martin Nöllenburg, editors, *Graph Drawing and Network Visualization – 24th International Symposium, GD 2016, Athens, Greece, September 19-21, 2016, Revised Selected Papers*, volume 9801 of *Lecture Notes in Computer Science*, pages 468–481. Springer, 2016.

14 Radoslav Fulek, Michael J. Pelsmajer, and Marcus Schaefer. Hanani-Tutte for radial planarity. *J. Graph Algorithms Appl.*, 21(1):135–154, 2017.

15 Radoslav Fulek, Michael J. Pelsmajer, Marcus Schaefer, and Daniel Štefankovič. Adjacent crossings do matter. *Journal of Graph Algorithms and Applications*, 16(3):759–782, 2012.

16 Carsten Gutwenger, Petra Mutzel, and Marcus Schaefer. Practical experience with Hanani-Tutte for testing c-planarity. In Catherine C. McGeoch and Ulrich Meyer, editors, *2014 Proceedings of the Sixteenth Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 86–97. SIAM, 2014. doi:10.1137/1.9781611973198.9.

17 Balázs Keszegh. Coloring intersection hypergraphs of pseudo-disks. *Discrete & Computational Geometry*, pages 1–23, 2019.

18 Jan Kynčl. Reply to "issue update: in graph theory, different definitions of edge crossing numbers – impact on applications?". https://mathoverflow.net/questions/366765/issue-update-in-graph-theory-different-definitions-of-edge-crossing-numbers (last accessed 8/6/2020), 2020.

19 Martin Loebl and Gregor Masbaum. On the optimality of the Arf invariant formula for graph polynomials. *Adv. Math.*, 226(1):332–349, 2011.

20 Bojan Mohar. The genus crossing number. *Ars Math. Contemp.*, 2(2):157–162, 2009.

21 Bojan Mohar and Carsten Thomassen. *Graphs on surfaces*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, MD, 2001.

22 Serguei Norine. Pfaffian graphs, t-joins and crossing numbers. *Combinatorica*, 28(1):89–98, 2008.

23 János Pach and Micha Sharir. On the boundary of the union of planar convex sets. *Discrete Comput. Geom.*, 21(3):321–328, 1999.

24 János Pach and Géza Tóth. Thirteen problems on crossing numbers. *Geombinatorics*, 9(4):194–207, 2000.

25 Michael J. Pelsmajer, Marcus Schaefer, and Despina Stasi. Strong Hanani-Tutte on the projective plane. *SIAM J. Discrete Math.*, 23(3):1317–1323, 2009.

26 Michael J. Pelsmajer, Marcus Schaefer, and Daniel Štefankovič. Removing even crossings. *J. Combin. Theory Ser. B*, 97(4):489–500, 2007.

27 Michael J. Pelsmajer, Marcus Schaefer, and Daniel Štefankovič. Odd crossing number and crossing number are not the same. *Discrete Comput. Geom.*, 39(1):442–454, 2008.

28 Michael J. Pelsmajer, Marcus Schaefer, and Daniel Štefankovič. Removing even crossings on surfaces. *European J. Combin.*, 30(7):1704–1717, 2009.

29 Michael J. Pelsmajer, Marcus Schaefer, and Daniel Štefankovič. Removing independently even crossings. *SIAM Journal on Discrete Mathematics*, 24(2):379–393, 2010.

30 Marcus Schaefer. The graph crossing number and its variants: A survey. *The Electronic Journal of Combinatorics*, 20:1–90, 2013. Dynamic Survey, #DS21, last updated September 2020.

31 Marcus Schaefer. Toward a theory of planarity: Hanani-Tutte and planarity variants. *Journal of Graph Algortihms and Applications*, 17(4):367–440, 2013.

32 Marcus Schaefer. Hanani-Tutte and related results. In I. Bárány, K. J. Böröczky, G. Fejes Tóth, and J. Pach, editors, *Geometry—Intuitive, Discrete, and Convex—A Tribute to László Fejes Tóth*, volume 24 of *Bolyai Society Mathematical Studies*. Springer, Berlin, 2014.

33 Shakhar Smorodinsky and Micha Sharir. Selecting points that are heavily covered by pseudo-circles, spheres or rectangles. *Combin. Probab. Comput.*, 13(3):389–411, 2004.

34 Glenn Tesler. Matchings in graphs on non-orientable surfaces. *Journal of Combinatorial Theory, Series B*, 78(2):198–231, 2000.

35 William T. Tutte. Toward a theory of crossing numbers. *J. Combinatorial Theory*, 8:45–53, 1970.

36 S. Whitesides and R. Zhao. K-admissible collections of Jordan curves and offsets of circular arc figures. *Technical Report SOCS 90.08, McGill University*, 1990.

Improved Approximation Algorithms for 2-Dimensional Knapsack: Packing into Multiple L-Shapes, Spirals, and More

Waldo Gálvez
Department of Computer Science, TU München, Germany

Fabrizio Grandoni
IDSIA, USI-SUPSI, Lugano, Switzerland

Arindam Khan
Department of Computer Science and Automation, Indian Institute of Science, Bangalore, India

Diego Ramírez-Romero
Department of Mathematical Engineering, Universidad de Chile, Santiago, Chile

Andreas Wiese
Department of Industrial Engineering and Center for Mathematical Modeling, Universidad de Chile, Santiago, Chile

Abstract

In the 2-Dimensional Knapsack problem (2DK) we are given a square knapsack and a collection of n rectangular items with integer sizes and profits. Our goal is to find the most profitable subset of items that can be packed non-overlappingly into the knapsack. The currently best known polynomial-time approximation factor for 2DK is $17/9 + \varepsilon < 1.89$ and there is a $(3/2 + \varepsilon)$-approximation algorithm if we are allowed to rotate items by 90 degrees [Gálvez et al., FOCS 2017]. In this paper, we give $(4/3 + \varepsilon)$-approximation algorithms in polynomial time for both cases, assuming that all input data are integers polynomially bounded in n.

Gálvez et al.'s algorithm for 2DK partitions the knapsack into a constant number of rectangular regions plus *one* L-shaped region and packs items into those in a structured way. We generalize this approach by allowing up to a *constant* number of *more general* regions that can have the shape of an L, a U, a Z, a spiral, and more, and therefore obtain an improved approximation ratio. In particular, we present an algorithm that computes the essentially optimal structured packing into these regions.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis

Keywords and phrases Approximation algorithms, two-dimensional knapsack, geometric packing

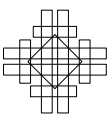
Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.39

Related Version *Full Version*: https://arxiv.org/abs/2103.10406

Funding *Waldo Gálvez*: Supported by the European Research Council, Grant Agreement No. 691672, project APEG.
Fabrizio Grandoni: Partially supported by the SNSF Excellence Grant 200020B_182865/1.
Andreas Wiese: Partially supported by the ANID Fondecyt Regular grant 1200173.

1 Introduction

The 2-Dimensional (Geometric) Knapsack problem (2DK) is a natural geometric generalization of the fundamental (one-dimensional) Knapsack problem. In 2DK we are given a set I of n items i which are axis-parallel rectangles specified by their width $w(i) \in \mathbb{N}$, height $h(i) \in \mathbb{N}$, and profit $p(i) \in \mathbb{N}$. Furthermore, we are given an axis-parallel square knapsack $K = [0, N] \times [0, N]$ for some $N \in \mathbb{N}$. The goal is to select a subset $I' \subseteq I$ of maximum total profit $p(I') := \sum_{i \in I'} p(i)$ that can be placed non-overlappingly inside K. Formally, for



each $i \in I'$ we have to define a pair $(lc(i), bc(i))$ that specifies the left and bottom coordinates of i, respectively, such that i is placed inside K as $R(i) := (lc(i), bc(i)) \times (lc(i)+w(i), bc(i)+h(i))$; we require that $R(i) \subseteq K$ and also that for any two $i, j \in I'$ it holds that $R(i) \cap R(j) = \emptyset$. We will consider also the case *with rotations*, in which each item $i \in I$ can be rotated by 90 degrees, i.e., i can be replaced by a rectangle with width $h(i)$, height $w(i)$ (and profit $p(i)$). In the cardinality (or unweighted) setting of the problem all profits are 1.

2DK has several applications. For example, the rectangles can model banners out of which one wants to place the most profitable subset on a website or an advertisement board. Also, they can model pieces that one wants to cut out of some raw material like wood or steel. In addition, there are scheduling settings in which jobs need a consecutive amount of some given resource (e.g., a frequency bandwidth) for some amount of time; thus, each job can be modeled via a rectangle.

Most algorithms for 2DK and related problems work as follows: they guess a partition of the knapsack into $O_\varepsilon(1)$ rectangular boxes, for some small constant $\varepsilon > 0$. Inside each box the items are packed greedily using the Next-Fit-Decreasing-Height algorithm [18], or even simpler by stacking items on top of each other or next to each other. Implicitly, Jansen and Zhang use this strategy to obtain a $(2+\varepsilon)$-approximation algorithm for 2DK [38, 39]. The same approach, however using $(\log N)^{O_\varepsilon(1)}$ boxes, is used in a QPTAS which assumes that N is quasi-polynomially bounded in n [4]. Finding a PTAS for 2DK or ruling it out is a major open problem in the area. This question is also open if N is polynomially bounded in n, i.e., for pseudo-polynomial time algorithms.

One might wonder whether a PTAS can be constructed using $O_\varepsilon(1)$ boxes only. Unfortunately, as observed in [24], essentially no better approximation ratio than 2 is achievable in this way. Hence a different type of packing is needed to breach this approximation barrier (in polynomial time). This was recently achieved by Gálvez et al. [24], where the authors pack the items into $O_\varepsilon(1)$ boxes and additionally one container with the shape of an L (which is packed with an ad-hoc, more complex algorithm).

This yields an approximation ratio of $17/9 + \varepsilon < 1.89$ (and $558/325 + \varepsilon < 1.72$ in the unweighted case). The authors also present $(3/2+\varepsilon)$- and $(4/3+\varepsilon)$-approximation algorithms for 2DK with rotations in the weighted and unweighted case, respectively.

Gálvez et al. [24] pose as an open problem how to efficiently pack items into a *constant* number of L-shaped containers, and observe that this would lead to improved approximation algorithms for 2DK. This problem was open even for just two L-shaped containers and using pseudo-polynomial time $(nN)^{O_\varepsilon(1)}$. In this paper we solve (a generalization of) this problem, and hence obtain an improved approximation ratio.

1.1 Our contribution

In this paper, we present a $(4/3+\varepsilon)$-approximation algorithm with a (pseudo-polynomial) running time of $(nN)^{O_\varepsilon(1)}$ for (weighted) 2DK. We also achieve improved $(4/3+\varepsilon)$- and $(5/4+\varepsilon)$-approximation algorithms for 2DK *with rotations* in the weighted and unweighted case, resp., with the same running time. See Table 1 for an overview of our and previous results in the respective settings.

Our algorithms use $O_\varepsilon(1)$ boxes and, in addition, rather than one single L-shaped container as in [24], a combination of $O_\varepsilon(1)$ containers with the shape of an L or even more complicated shapes. The latter are intuitively thin corridors with the property that if we traverse them, we change the orientation at the turns (i.e., clockwise or counter-clockwise) at most once. For example, they can have figuratively the shape of a U, a Z, or a spiral (see Figure 1). Since they are thin, they help us to distinguish the parts of K that are used by items that are wide and thin (horizontal items) and items that are high and narrow (vertical items). The interaction of these types of items is a major difficulty in 2DK.

Table 1 A summary of our approximation ratios compared to the best known results with running time $(nN)^{O_\varepsilon(1)}$ for the respective settings.

Setting		Known result	Our result
Without rotations	Weighted	$17/9+\varepsilon < 1.89$ [24]	$4/3+\varepsilon$
	Unweighted	$558/325+\varepsilon < 1.72$ [24]	$4/3+\varepsilon$
With rotations	Weighted	$3/2+\varepsilon$ [24]	$4/3+\varepsilon$
	Unweighted	$4/3+\varepsilon$ [24]	$5/4+\varepsilon$

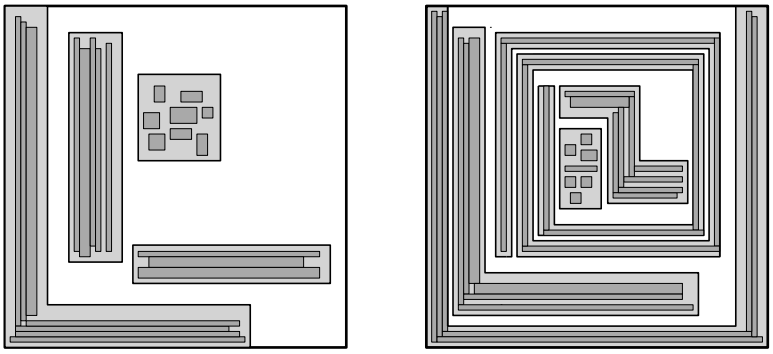


Figure 1 Left: a packing based on a single L-shaped container and boxes as it was used in previous work. Right: A packing based on a box and cooridors with the shapes of an L, a U, a Z, and a spiral, as we use them in our algorithms.

By standard arguments (in particular building upon the corridor decomposition in [2]), it is not hard to show that we can partition K into a constant number of boxes and corridors of the allowed types, so that there exists a feasible packing of a $(4/3+\varepsilon)$-approximate solution into them. The non-trivial part is how to efficiently pack items into our corridors. Here we cannot exploit the L-packing algorithm in [24]. Indeed, the latter algorithm does not seem to generalize even to two L-shaped corridors (even if $N = n^{O(1)}$), while we need to handle $\Theta_\varepsilon(1)$ corridors with possibly even more general shapes.

Our strategy is to partition each of our corridors into $O_\varepsilon(\log N)$ rectangular boxes. Using the properties of their shapes, we show that this is indeed possible by losing only a factor of $1+\varepsilon$ in the approximation guarantee. Guessing these boxes explicitly would take $N^{O_\varepsilon(\log N)}$ time which is too slow. Instead, we show how to guess the *sizes* of almost all of the boxes in time $(nN)^{O_\varepsilon(1)}$. Then, we place them into the corridors in polynomial time using a dynamic program based on color-coding, using that in total there are only $O_\varepsilon(\log N)$ boxes to place.

We remark that the above approach compromises between corridor shapes that are general enough to allow for $(4/3+\varepsilon)$ -approximate packings, and at the same time simple enough so that we can partition them into $O_\varepsilon(\log N)$ boxes that we can essentially guess in time $(nN)^{O_\varepsilon(1)}$. It is not clear how to extend our algorithm to $\Theta(\log^{1/\varepsilon} N)$, or just even $\Theta(\log^2 N)$ such boxes. However, this would allow us to exploit corridors of more general shapes (say W-shaped), hence achieving better approximation ratios. We leave this as an interesting open problem.

1.2 Related Work

The QPTAS in [4] (though with some restrictions on N) suggests that 2DK most likely admits a PTAS. This is already known for some relevant special cases: if the profit of each item equals its area [7], if the size of the knapsack can be slightly increased (resource augmentation) [22, 35], if all items are relatively small [21], or squares [31, 36].

One might consider packing geometric objects other than rectangles. In particular, there are constant approximation algorithms for packing triangles and also arbitrary convex polygons under resource augmentation, both assuming that arbitrary rotations are allowed [46]. Also, for circles a $(1+\varepsilon)$-approximation is known under resource augmentation in one dimension if the profit of each circle equals its area [45]. One can consider natural generalizations of 2DK to a higher number of dimensions. In particular, the 3-dimensional case, 3DK, has applications like packing containers into a ship or cargo into a truck. 3DK is known to be APX-hard [14], and constant approximation algorithms are known [19, 27]. Khan et al. [44] have given a $(2+\varepsilon)$-approximation algorithm for a generalization of 2DK, which generalizes geometric packing and vector packing.

A parameterized version of 2DK for rectangles (where the parameter is the number k of packed items) is studied in [26]. The authors show that the problem is $W[1]$-hard (both with and without rotations). Furthermore, they provide an FPT $(1+\varepsilon)$-approximation for the case with rotations. Achieving a similar result for the case without rotations is open.

A packing is called a guillotine packing if all rectangles can be separated by a sequence of end-to-end (guillotine) cuts [43]. Abed et al. [1] have given QPTAS for 2DK satisfying guillotine packing constraints, assuming the input data is quasi-polynomially bounded. Recently, Khan et al. [42] have shown a pseudo polynomial-time approximation scheme for 2DK satisfying guillotine packing constraints.

In the 2-Dimensional Bin Packing problem we are given a collection of items similarly to 2DK, and copies of the same square knapsack (the bins). Our goal is to pack *all* the items using the smallest possible number of bins. The best known (asymptotic) result for this problem is due to Bansal and Khan [9]: they achieve a 1.405 approximation based on a configuration-LP. This improves a series of previous results [8, 10, 16, 35, 40].

Another closely related problem is Strip Packing. Informally, we are given a knapsack of width N and infinite height, and we wish to pack *all* items so that the topmost coordinate is as small as possible. This problem admits a $(5/3+\varepsilon)$-approximation [28] (improving on [6, 18, 29, 49, 50, 51]) in the general case, $(3/2+\varepsilon)$-approximation [23] when none of the items are large, and it is NP-hard to approximate it below a factor $3/2$ by a simple reduction from Partition. However, strictly better approximation ratios can be achieved in pseudo-polynomial time $(Nn)^{O_\varepsilon(1)}$ [25, 34, 47], being $5/4+\varepsilon$ essentially the best possible ratio achievable in this setting [30, 33]. This shows that pseudo-polynomial time can make the difference for rectangle packing problems. It would be interesting to understand whether the techniques in our paper (as well as those in [4]) can be strengthened so as to run in polynomial time for arbitrary N. Strip Packing was also studied in the asymptotic setting [35, 40] and in the case with rotations [37].

Another related problem is the Independent Set of Rectangles problem: here we are given a collection of axis-parallel rectangles embedded in the plane, and we need to find a maximum cardinality/weight subset of non-overlapping rectangles [2, 3, 11, 17]. The problem has also been studied for squares, disks, and pseudo-disks, see e.g., [20, 32, 12].

We refer the readers to [15, 41] for surveys on geometric packing problems.

2 Preliminaries

We start with a classification of the input items according to their heights and widths. Let $\varepsilon > 0$. For two constants $1 \geq \varepsilon_{large} > \varepsilon_{small} > 0$ to be defined later, we classify an item i as:

- *small*: if $h(i), w(i) \leq \varepsilon_{small} N$;
- *large*: if $h(i), w(i) > \varepsilon_{large} N$;

- *horizontal*: if $w(i) > \varepsilon_{large} N$ and $h(i) \leq \varepsilon_{small} N$;
- *vertical*: if $h(i) > \varepsilon_{large} N$ and $w(i) \leq \varepsilon_{small} N$;
- *intermediate*: otherwise, i.e., the length of at least one edge is in $(\varepsilon_{small} N, \varepsilon_{large} N]$.

We call *skewed* the items that are either horizontal or vertical. We let I_{small}, I_{large}, I_{hor}, I_{ver}, I_{skew}, and I_{int} be the items which are small, large, horizontal, vertical, skewed, and intermediate, respectively. The corresponding intersection with the optimal solution OPT defines the sets OPT_{small}, OPT_{large}, OPT_{hor}, OPT_{ver}, OPT_{skew} and OPT_{int}, respectively.

In order to describe our main ideas, we will start by considering the cardinality case of the problem (i.e., $p(i) = 1$ for each item $i \in I$) without rotations. In Sections 3, 4, and 5, we will present a simplified algorithm that yields a $1.6 + \varepsilon$ approximation.

Notice that the optimum solution can contain at most $1/\varepsilon_{large}^2$ large items. Thus, unless $|OPT| \leq 1/(\varepsilon \cdot \varepsilon_{large}^2)$ (in which case we can solve the problem optimally in time $n^{O(1/(\varepsilon \cdot \varepsilon_{large}^2))}$ by complete enumeration), we can drop all large items by losing only a factor of $1 + \varepsilon$ in the approximation. Similarly, by standard shifting techniques (e.g. Lemma 2.1 in [24]), when defining ε_{large} and ε_{small} one can ensure that the intermediate items can be neglected by losing only a factor of $1 + \varepsilon$ in the approximation ratio (while maintaining that ε_{large} and ε_{small} are lower-bounded by some constant depending only on ε).

Hence, w.l.o.g. we can assume that all items are small or skewed. It is possible to deal with small items by standard techniques from the literature (e.g. Section 6.2.1 in [24]), however this would make our exposition much more technical without introducing substantially new ideas. Hence, for the sake of simplicity, we will assume that there are no small items, i.e., all items are skewed. We remark that, even with the mentioned restrictions, the problem is far from being trivial. In particular, the best known approximation for the considered setting is $\frac{558}{325} + \varepsilon \approx 1.72$ [24]. Our simplified algorithm has a better approximation ratio $1.6 + \varepsilon$, and it is also substantially simpler.

For the main result, which is a $(4/3 + \varepsilon)$-approximation algorithms for the general case with and without rotations, please refer to the full version of this work.

3 Partition into LU-corridors

Our strategy is to partition the knapsack into $O_\varepsilon(1)$ thin corridors, each having the shape of an L or a U, such that there exists a $(1.6 + \varepsilon)$-approximate solution in which each item is contained in one of these corridors (see Figure 2). In this section, we will make this precise and show that such a partition indeed exists. Our algorithm will then guess this partition in polynomial time. In Sections 4 and 5 we will show how to find the corresponding solution afterwards efficiently.

Intuitively, a *path corridor* is a polygon inside K that describes a path of width at most $\varepsilon \cdot \varepsilon_{large}$ and that is allowed to have bends, see Figure 3. Formally, it is a simple rectilinear polygon within K with $2k$ edges $e_0, \ldots, e_{2k-1}$ for some integer $k \geq 2$, such that for each pair of horizontal (resp., vertical) edges e_i, e_{2k-i}, $i \in \{1, ..., k-1\}$ there exists a vertical (resp., horizontal) line segment ℓ_i of length less than $\varepsilon \cdot \varepsilon_{large}$ such that both e_i and e_{2k-i} intersect ℓ_i and ℓ_i does not intersect any other edge, and require e_i and e_{2k-i} to have length at least $\varepsilon_{large}/2$. Note that e_0 and e_k are not required to satisfy these properties. We say that such a path corridor C has $s(C) := k - 1$ *subcorridors.* We say that a *box* is a path corridor with only one subcorridor, i.e., it is simply a rectangle.

Similarly, a *cycle corridor* is intuitively a path corridor in which the start and end point of the path coincide (see Figure 3). Formally, we define it to be a face bounded by two simple non-intersecting rectilinear polygons defined by edges $e_0, e_1, \ldots, e_{k-1}$ and $e'_0, e'_1, \ldots, e'_{k-1}$,

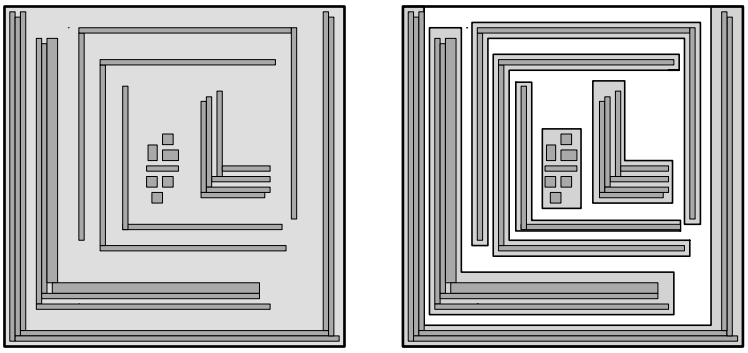


Figure 2 Left: a packing of horizontal, vertical and small items into the knapsack. Right: a decomposition of the knapsack into box-shaped, L-shaped and U-shaped corridors which contain the previous items.

each of them of length at least $\varepsilon_{large}/2$, such that the second polygon is contained in the first one, and for each pair of corresponding horizontal (vertical) edges e_i, e'_i for $i \in \{0, \ldots, k-1\}$ there is a vertical (horizontal, respectively) line segment ℓ_i of length less than $\varepsilon \cdot \varepsilon_{large}$ such that both edges e_i and e'_i intersect ℓ_i and ℓ_i does not intersect any other edge of the cycle corridor. We say that the resulting cycle corridor C has $s(C) := k$ subcorridors.

We use a result from [2] that implies directly that there there exists a partition of K into $O_\varepsilon(1)$ corridors and a near-optimal solution in which each item is contained in some corridor.

▶ **Lemma 1** ([2]). *There exists a solution $\overline{OPT} \subseteq OPT$ with $|OPT| \leq (1+\varepsilon)\left|\overline{OPT}\right|$ and a partition of K into a set of corridors $\bar{\mathcal{C}}$ with $|\bar{\mathcal{C}}| \leq O_\varepsilon(1)$ where each corridor $C \in \bar{\mathcal{C}}$ has at most $1/\varepsilon$ subcorridors and each item $i \in \overline{OPT}$ is contained in a corridor of $\mathcal{C}$.*

Algorithmically, we could guess $\bar{\mathcal{C}}$ in time $N^{O_\varepsilon(1)}$ and then try to compute a profitable solution in which each item is contained in one corridor in $\bar{\mathcal{C}}$, i.e., mimicking $\overline{OPT}$. However, it is not clear how to compute such a solution in polynomial time. Therefore, we partition $\bar{\mathcal{C}}$ further into a set of smaller corridors $\mathcal{C}$ such that each resulting corridor has the shape of an L or a U. We will ensure that there exists a $(1.6+\varepsilon)$-approximate solution in which each item is contained in one corridor of $\mathcal{C}$. Since the corridors in $\mathcal{C}$ are simpler than the corridors in $\bar{\mathcal{C}}$, we will be able to compute in polynomial time essentially the most profitable solution in which each item is contained in a corridor of $\mathcal{C}$ (see in Sections 4 and 5).

We say that a path corridor C is an *L-corridor* if C has exactly two subcorridors, and then figuratively it has the shape of an L (see Figure 3). Intuitively, we define that a path corridor is a U-corridor if it has the shape of a U. Formally, let C be a a path corridor with exactly three subcorridors, and hence C is defined via edges $e_0, ..., e_7$. Assume w.l.o.g. that e_2 and e_6 are horizontal. We project e_2 and e_6 on the x-axis, let I_2 and I_6 be the resulting intervals. Then C is a *U-corridor* if $I_2 \subseteq I_6$ or $I_6 \subseteq I_2$.

In order to partition $\bar{\mathcal{C}}$, we first define a partition of each path/cycle corridor $C \in \bar{\mathcal{C}}$ into $s(C)$ subcorridors (see Figure 3). We say that a *subcorridor* of a corridor C is a simple polygon $P \subseteq C$ whose boundary consists of two parallel edges (in most cases these will be edges of C) and of two monotone axis-parallel curves, i.e., sets of axis-parallel line segments such that either for any two points $(x_1, y_1), (x_2, y_2) \in P$ where $x_1 < x_2$ we have $y_1 \leq y_2$, or for any such two points we have $y_1 \geq y_2$. We require that each vertex of P has integral coordinates. Given a corridor C and a set of non-overlapping items I' inside C, we say that a partition of C into a set of subcorridors is *nice for* I' if each subcorridor either intersects only items from $I' \cap I_{hor}$ or only items from from $I' \cap I_{ver}$.

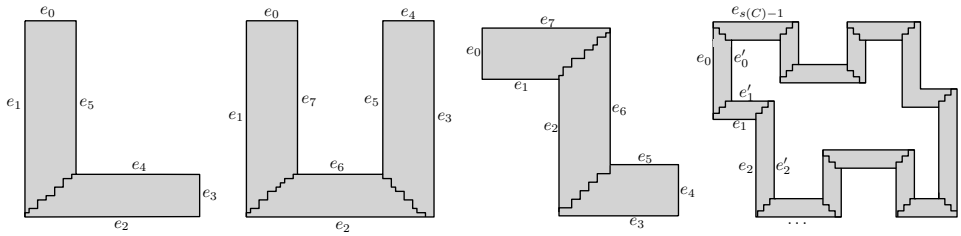


Figure 3 An L-corridor, a U-corridor, another path corridor with two bends, and a cycle corridor. The monotone axis-parallel curves indicate the boundaries of the subcorridors.

▶ **Lemma 2** ([4], Lemma 2.4). *Let C be a path/cycle corridor containing a set of items I'. There is a partition of C into $s(C)$ subcorridors that is nice for I'.*

Now we take each path corridor $C \in \bar{\mathcal{C}}$ and delete the items in every third subcorridor, starting with the α-th subcorridor for some offset $\alpha \in \{1, 2, 3\}$. Then we can divide C into L-corridors such that each remaining item is contained in one of these L-corridors. Each item $i \in \overline{OPT}$ contained in C is deleted only for one choice of α, and hence there is a choice for α such that we lose at most one third of the profit due to this step. Now consider a cycle corridor $C \in \bar{\mathcal{C}}$. Note that $s(C)$ is even and $s(C) \geq 4$. If $s(C) = 4$ (i.e., C is a ring), we delete the items in one of its four subcorridors, losing at most one quarter of the profit, and obtain a U-corridor. If $s(C) \geq 6$ and $s(C)$ is divisible by 3 we do the same operation as for path corridors, losing at most one third of the profit. For all other values of $s(C)$ we might lose a larger factor since then $s(C)$ is not divisible by 3 and P_0 and $P_{s(C)-1}$ are adjacent, e.g., if $s(C) = 8$. However, a case distinction shows that we can still partition C into L- and U-corridors while decreasing the profit at most by a factor of 1.6.

▶ **Lemma 3.** *There exists a solution $OPT' \subseteq OPT$ with $|OPT| \leq (1.6 + \varepsilon)|OPT'|$ and a partition of K into a set $\mathcal{C}$ of $O_\varepsilon(1)$ L- and U-corridors such that each item $i \in OPT'$ is contained in one corridor of $\mathcal{C}$.*

The first step in our algorithm is to guess $\mathcal{C}$ which can be done in time $N^{O_\varepsilon(1)}$. The next step is to compute a solution with at least $(1 - \varepsilon)|OPT'|$ items in which each item is contained in one corridor of $\mathcal{C}$. For this, we consider two cases separately, which are intuitively the case that $|OPT'| \geq \Omega_\varepsilon(\log N)$ and $|OPT'| \leq O_\varepsilon(\log N)$, and they are treated in Sections 4 and 5, respectively.

4 Packing via guessing slices

In this section, we assume that $|OPT'| > c_\varepsilon \cdot \log N$ for some constant c_ε to be defined later. We describe an algorithm that computes a solution of size $(1 - \varepsilon)|OPT'|$ such that each item of this solution is contained in a corridor in $\mathcal{C}$.

First, we group the items into $O_\varepsilon(\log N)$ groups where we group the items in I_{hor} according to their heights and the items in I_{ver} according to their widths. Formally, for each $\ell \in \{0, ..., \lfloor \log_{1+\varepsilon} N \rfloor\}$ we define $I^{(\ell)}_{hor} := \{i \in I_{hor} | h(i) \in [(1+\varepsilon)^\ell, (1+\varepsilon)^{\ell+1})\}$ and $I^{(\ell)}_{ver} := \{i \in I_{ver} | w(i) \in [(1+\varepsilon)^\ell, (1+\varepsilon)^{\ell+1})\}$. So intuitively, for each ℓ the items in $I^{(\ell)}_{hor}$ essentially all have the same height and the items in $I^{(\ell)}_{ver}$ essentially all have the same width. Now, for the groups $I^{(\ell)}_{hor}$, $I^{(\ell)}_{ver}$ we guess estimates $\text{opt}^{(\ell)}_{hor}, \text{opt}^{(\ell)}_{ver}$ for $|I^{(\ell)}_{hor} \cap OPT'|$, $|I^{(\ell)}_{ver} \cap OPT'|$, respectively. Even though there can be $\Theta_\varepsilon(\log N)$ of these groups and for each guessed value there are potentially $\Omega(n)$ options, we guess the estimates for *all* groups in parallel in time $(nN)^{O_\varepsilon(1)}$, adapting a technique from [13].

▶ **Lemma 4.** *In time* $(nN)^{O_\varepsilon(1)}$ *we can guess the values for all pairs* $\mathrm{opt}_{hor}^{(\ell)}, \mathrm{opt}_{ver}^{(\ell)}$ *with* $\ell \in \{0, ..., \lfloor \log_{1+\varepsilon} N \rfloor\}$ *such that*

- $\sum_\ell \mathrm{opt}_{hor}^{(\ell)} + \mathrm{opt}_{ver}^{(\ell)} \geq (1-\varepsilon)|OPT'|$ *and*
- $\mathrm{opt}_{hor}^{(\ell)} \leq |OPT' \cap I_{hor}^{(\ell)}|$ *and* $\mathrm{opt}_{ver}^{(\ell)} \leq |OPT' \cap I_{hor}^{(\ell)}|$ *for each* $\ell \in \{0, ..., \lfloor \log_{1+\varepsilon} N \rfloor\}$.

Proof. First, we guess $|OPT'|$ for which there are only n options. We show now how to guess in time $N^{O_\varepsilon(1)}$ the feasible values for $\mathrm{opt}_{hor}^{(\ell)}$; a symmetric argument holds for $\mathrm{opt}_{ver}^{(\ell)}$. Let us define each $\mathrm{opt}_{hor}^{(\ell)}$ as the largest integer of the form $k_{hor}^{(\ell)} \cdot \frac{\varepsilon}{4\log_{1+\varepsilon} N}|OPT'|$ which is upper bounded by $|OPT' \cap I_{hor}^{(\ell)}|$, where $k_{hor}^{(\ell)}$ is a non-negative integer. Notice that trivially $\sum_\ell k_{hor}^{(\ell)} \leq \frac{4\log_{1+\varepsilon} N}{\varepsilon}$. We encode all such values $k_{hor}^{(\ell)}$ as a single binary string as follows: we represent each $k_{hor}^{(\ell)}$ as a string of $k_{hor}^{(\ell)}$ 0-bits followed by one 1-bit, and then chain such strings according to the index ℓ. The final bit string encodes the solution. Notice that this string contains at most $\log_{1+\varepsilon} N + 1 + \sum_\ell k_{hor}^{(\ell)} = O(\frac{\log_{1+\varepsilon} N}{\varepsilon})$ bits, hence we can guess it in time $N^{O_\varepsilon(1)}$. The claim follows since

$$|OPT'| - \sum_\ell \left(\mathrm{opt}_{hor}^{(\ell)} + \mathrm{opt}_{ver}^{(\ell)}\right) \leq 2(\log_{1+\varepsilon} N + 1) \cdot \frac{\varepsilon}{4\log_{1+\varepsilon} N}|OPT'| \leq \varepsilon |OPT'|. \qquad \blacktriangleleft$$

Definition of slices. Next, for each group $I_{hor}^{(\ell)}$ we define slices that together are essentially as profitable as the items in $OPT' \cap I_{hor}^{(\ell)}$. We first order the items in $I_{hor}^{(\ell)}$ non-decreasingly by width and select the first $\frac{1}{1+\varepsilon}\mathrm{opt}_{hor}^{(\ell)}$ items. One can show easily that their total height is at most $h(OPT' \cap I_{hor}^{(\ell)})$. Let i be one of these items. Intuitively, we slice i horizontally into slices of height 1. Formally, for i we introduce $h(i)$ items of height 1 and profit $1/(1+\varepsilon)^\ell$ each. Let $\hat{I}_{hor}^{(\ell)}$ denote the resulting set of slices. We do this procedure for each ℓ and a symmetric procedure for the group $I_{ver}^{(\ell)}$ for each ℓ, resulting in a set of slices $\hat{I}_{ver}^{(\ell)}$.

▶ **Lemma 5.** *It is possible to place the slices in* $\left\{\hat{I}_{hor}^{(\ell)}, \hat{I}_{ver}^{(\ell)}\right\}_\ell$ *non-overlappingly inside* K *such that each slice is contained in some corridor in* $\mathcal{C}$*. Also, we have that* $\sum_\ell \left(p(\hat{I}_{hor}^{(\ell)}) + p(\hat{I}_{ver}^{(\ell)})\right) \geq \frac{1}{1+O(\varepsilon)}|OPT'|$.

Proof sketch. Let $\ell \in \{0, ..., \lfloor \log_{1+\varepsilon} N \rfloor\}$. Recall that $\mathrm{opt}_{hor}^{(\ell)} \leq |OPT' \cap I_{hor}^{(\ell)}|$, all items in $I_{hor}^{(\ell)}$ have the same height (up to a factor of $1+\varepsilon$), and we selected the $\frac{1}{1+\varepsilon}\mathrm{opt}_{hor}^{(\ell)}$ items in $I_{hor}^{(\ell)}$ of minimum width. Using this, one can show that the slices in $\hat{I}_{hor}^{(\ell)}$ fit into the space that is occupied by the items in $OPT' \cap I_{hor}^{(\ell)}$ in OPT'. Also, $\frac{1}{1+O(\varepsilon)}h(OPT' \cap I_{hor}^{(\ell)}) \leq |\hat{I}_{hor}^{(\ell)}| \leq h(OPT' \cap I_{hor}^{(\ell)})$ and each slice in $\hat{I}_{hor}^{(\ell)}$ has a profit of $1/(1+\varepsilon)^\ell$. Using this for each ℓ and a similar statement for the vertical items, one can prove the second claim of the lemma. $\blacktriangleleft$

Next, for each $\ell \in \{0, ..., \lfloor \log_{1+\varepsilon} N \rfloor\}$ we round the widths of the slices in $\hat{I}_{hor}^{(\ell)}$ via linear grouping such that they have at most $1/\varepsilon$ different widths and we lose at most a factor of $1 + O(\varepsilon)$ in their profit due to this rounding. Formally, we sort the slices in $\hat{I}_{hor}^{(\ell)}$ non-increasingly by width and then partition them into $1/\varepsilon + 1$ groups such that each group contains $\left\lceil \frac{1}{1/\varepsilon+1}|\hat{I}_{hor}^{(\ell)}| \right\rceil$ slices (apart from possibly the last group which might contain fewer slices). Let $\hat{I}_{hor}^{(\ell)} = \hat{I}_{hor,1}^{(\ell)} \dot{\cup} ... \dot{\cup} \hat{I}_{hor,1/\varepsilon+1}^{(\ell)}$ denote the resulting partition. We drop the slices in $\hat{I}_{hor,1}^{(\ell)}$ (whose total profit is at most $\varepsilon \cdot p(\hat{I}_{hor}^{(\ell)})$). Then, for each $j \in \{2, ..., 1/\varepsilon + 1\}$ we increase the width of the slices in $\hat{I}_{hor,j}^{(\ell)}$ to the width of the widest slice in $\hat{I}_{hor,j}^{(\ell)}$. By construction, the resulting slices have $1/\varepsilon$ different widths. Let $\tilde{I}_{hor}^{(\ell)}$ denote the resulting set and let $\tilde{I}_{hor}^{(\ell)} = \tilde{I}_{hor,1}^{(\ell)} \dot{\cup} ... \dot{\cup} \tilde{I}_{hor,1/\varepsilon}^{(\ell)}$ denote a partition of $\tilde{I}_{hor}^{(\ell)}$ according to the widths of the slices, i.e., for each $j \in \{1, ..., 1/\varepsilon\}$ the set $\tilde{I}_{hor,j}^{(\ell)}$ contains the rounded slices from $\hat{I}_{hor,j+1}^{(\ell)}$.

We do this procedure for each ℓ and a symmetric procedure for the group $I^{(\ell)}_{ver}$ for each ℓ.

▶ **Lemma 6.** *It is possible to place the slices in* $\left\{\tilde{I}^{(\ell)}_{hor,j}, \tilde{I}^{(\ell)}_{ver,j}\right\}_{\ell,j}$ *non-overlappingly inside the knapsack such that each slice is contained in some corridor in* $\mathcal{C}$*. Also, we have* $\sum_\ell \left(p(\tilde{I}^{(\ell)}_{hor}) + p(\tilde{I}^{(\ell)}_{ver})\right) \geq \frac{1}{1+\varepsilon} \sum_\ell \left(p(\hat{I}^{(\ell)}_{hor}) + p(\hat{I}^{(\ell)}_{ver})\right)$.

Proof sketch. For each $\ell \in \{0, ..., \lfloor \log_{1+\varepsilon} N \rfloor\}$ and $j \in \{2, ..., 1/\varepsilon + 1\}$, the slices in $\tilde{I}^{(\ell)}_{hor,j}$ fit into the space occupied by the slices in $\hat{I}^{(\ell)}_{hor,j-1}$. Also, for each ℓ we lost slices in $\hat{I}^{(\ell)}_{hor}$ with a total profit of at most $\varepsilon \cdot p(\hat{I}^{(\ell)}_{hor})$. A similar argumentation holds for the sets $I^{(\ell)}_{ver}$. ◀

We fix a partition of each corridor $C \in \mathcal{C}$ into subcorridors that is nice for the slices $\left\{\tilde{I}^{(\ell)}_{hor,j}, \tilde{I}^{(\ell)}_{ver,j}\right\}_{\ell,j}$. Note that we do not compute this partition explicitly but we use it for our analysis and as guidance for our algorithm. For each corridor $C \in \mathcal{C}$ or subcorridor S denote by $\tilde{I}(C)$ and $\tilde{I}(S)$ the slices assigned to C and S, resp., due to Lemma 6.

Structuring slices inside subcorridors. Consider a subcorridor S of a corridor $C \in \mathcal{C}$. The placement of the slices inside S due to Lemma 6 might be complicated. Instead, we would like to have a packing where the slices are packed *nicely*, i.e., they are stacked on top of each other if S is horizontal, and side by side if S is vertical (see Figure 4). This might not be possible to achieve exactly, but we construct something very similar. We prove that inside S we can place $O_\varepsilon(1)$ boxes and one subcorridor $S' \subseteq S$ (which we will call sub-subcorridor in order to distinguish it from the subcorridors) that are pairwise disjoint and such that inside them we can nicely place essentially all slices from $\tilde{I}(C)$ (see Figure 4). We can guess the placement of the $O_\varepsilon(1)$ boxes inside S. Unfortunately, we cannot guess S' directly, but we can guess the two edges that define the boundary of S' together with the two axis-parallel curves. We refer to them as the *edges of* S'. Note that they are horizontal if S (and hence S') is horizontal, and vertical otherwise. We construct S' such that the longer of these two edges is always also an edge of S (see Figure 4).

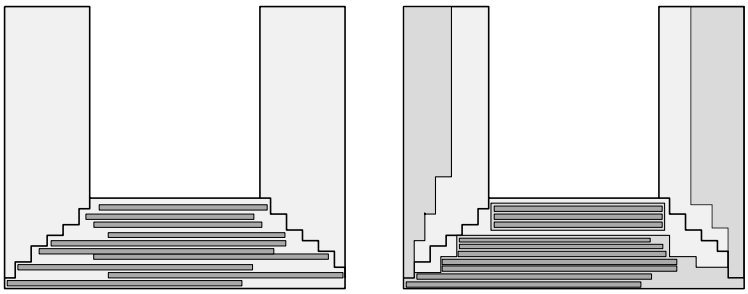


Figure 4 Left: a subcorridor with items packed inside it. Right: A partition of each subcorridor into $O_\varepsilon(1)$ boxes and a sub-subcorridor, all containing slices which are nicely packed.

▶ **Lemma 7.** *For each horizontal/vertical subcorridor* S *we can guess in time* $N^{O_\varepsilon(1)}$

- *the two edges of a horizontal/vertical sub-subcorridor* $S' \subseteq S$ *such that the longer edge of* S' *coincides with the longer edge of* S*,*
- $O_\varepsilon(1)$ *non-overlapping boxes* $\mathcal{B}(S)$ *inside* S *that are disjoint with* S'*,*

such that we can nicely place slices from $\tilde{I}(S)$ *with a total profit of* $(1-\varepsilon)p(\tilde{I}(S))$ *inside* S' *and the boxes* $\mathcal{B}(S)$*.*

We apply Lemma 7 to each subcorridor S of a corridor $C \in \mathcal{C}$. Let $\mathcal{S}$ and $\mathcal{B}$ denote the resulting set of sub-subcorridors and boxes, respectively. For each ℓ, each j, and each $F \in \mathcal{B} \cup \mathcal{S}$ denote by $\tilde{I}^{(\ell)}_{hor,j}(F)$ and $\tilde{I}^{(\ell)}_{ver,j}(F)$ the respective slices from $\tilde{I}^{(\ell)}_{hor,j}, \tilde{I}^{(\ell)}_{ver,j}$ in F, respectively. Using simple slice reorderings, we can prove the following lemma.

▶ **Lemma 8.** *There is a packing of the slices in* $\left\{\tilde{I}^{(\ell)}_{hor,j}(F), \tilde{I}^{(\ell)}_{ver,j}(F)\right\}_{j,\ell,F}$ *such that for each box or sub-subcorridor* $F \in \mathcal{B} \cup \mathcal{S}$ *we can assume w.l.o.g. that*

- *horizontal/vertical items inside* F *are ordered non-increasingly by width/height, starting at the longer edge of* F *if* F *is a sub-subcorridor, and starting at an arbitrary edge if* F *is a box; ties are broken according to the input items that the slices correspond to,*
- *any two adjacent horizontal/vertical slices of the same width/height are placed exactly on top of each other/side by side.*

Therefore, we can construct this packing of the slices if we knew the cardinality of $\tilde{I}^{(\ell)}_{hor,j}(F)$ and $\tilde{I}^{(\ell)}_{ver,j}(F)$ for each $F \in \mathcal{B} \cup \mathcal{S}$ and each ℓ and j. We guess this cardinality approximately in the following lemma for each F, ℓ and j in parallel.

▶ **Lemma 9.** *In time* $(nN)^{O_\varepsilon(1)}$ *we can guess values* $\operatorname{opt}^{(\ell)}_{hor,j}(F), \operatorname{opt}^{(\ell)}_{ver,j}(F)$ *for each* $\ell \in \{0, ..., \lfloor \log_{1+\varepsilon} N \rfloor\}$, $j \in \{1, ..., 1/\varepsilon\}$, $F \in \mathcal{B} \cup \mathcal{S}$ *such that*

- $\operatorname{opt}^{(\ell)}_{hor,j}(F) \leq \left|\tilde{I}^{(\ell)}_{hor,j}(F)\right|$ *and* $\operatorname{opt}^{(\ell)}_{ver,j}(F) \leq \left|\tilde{I}^{(\ell)}_{ver,j}(F)\right|$ *for each* ℓ, j, F *and*
- $\sum_{F \in \mathcal{B} \cup \mathcal{S}} \operatorname{opt}^{(\ell)}_{hor,j}(F) \geq (1-\varepsilon) \sum_{F \in \mathcal{B} \cup \mathcal{S}} \left|\tilde{I}^{(\ell)}_{hor,j}(F)\right|$ *and* $\sum_{F \in \mathcal{B} \cup \mathcal{S}} \operatorname{opt}^{(\ell)}_{ver,j}(F) \geq (1-\varepsilon) \sum_{F \in \mathcal{B} \cup \mathcal{S}} \left|\tilde{I}^{(\ell)}_{ver,j}(F)\right|$ *for each* ℓ, j.

Proof sketch. For each ℓ, j, and F we define $\operatorname{opt}^{(\ell)}_{hor,j}(F)$ to be the largest integral multiple of $\frac{\varepsilon}{|\mathcal{B}|+|\mathcal{S}|}\left|\tilde{I}^{(\ell)(F)}_{hor,j}\right|$ that is at most $\left|\tilde{I}^{(\ell)}_{hor,j}(F)\right|$ and note that for this value there are only $\frac{|\mathcal{B}|+|\mathcal{S}|}{\varepsilon} = O_\varepsilon(1)$ options. We define the values $\operatorname{opt}^{(\ell)}_{ver,j}(F)$ similarly. Since there are only $O_\varepsilon(\log nN)$ of these values altogether, we can guess all of them in time $2^{O_\varepsilon(\log nN)} = (nN)^{O_\varepsilon(1)}$. ◀

Placing slices inside subcorridors. Given the number of slices in each box and each sub-subcorridor due to Lemma 9, we compute a corresponding packing for the slices. Inside of each box we simply sort the slices by height or width, respectively, and then pack them in this order. For packing the slices inside the sub-subcorridors of a corridor C, recall that we do not know the precise sub-subcorridors, we know only the guessed edges due to Lemma 7. However, we can still find a packing for the slices inside of the sub-subcorridors of C. We start with the first sub-subcorridor S_1 of C, sort its slices by height or width, respectively (breaking ties according to the input items that the slices correspond to), and place them in this order, starting at the longer edge of S_1. When we do this, we push the slices as far as possible to the edge e_0. The resulting packing satisfies the properties of Lemma 8. If $s(C) \geq 2$ then we do the same procedure for the last sub-subcorridor $S_{s(C)}$ of C, and in particular we push its slices as far as possible to the edge $e_{s(C)}$. If $s(C) \in \{1, 2\}$ then we are done now. Otherwise $s(C) = 3$ since $s(C) \leq 3$ for each $C \in \mathcal{C}$ and the slices of the second sub-subcorridor S_2 are still not placed. We sort the slices as before and place them in this order, starting at the longer edge of S_2 and such that their placement satisfies the properties of Lemma 8. Since we had pushed the slices in S_1 and S_3 maximally to the edges e_0 and e_k, one can show that this is indeed possible.

Rounding slices. For each set $I^{(\ell)}_{hor}, I^{(\ell)}_{ver}$, their corresponding slices induce in total $O_\varepsilon(1)$ rectangular areas into which we assigned these slices: at most one for each of the $O_\varepsilon(1)$ sub-subcorridors and at most one for each of the $O_\varepsilon(1)$ boxes inside each of the $O_\varepsilon(1)$

subcorridors. For each ℓ we denote by $\mathcal{B}^{(\ell)}_{hor}, \mathcal{B}^{(\ell)}_{ver}$ these corresponding areas which are in fact boxes. Now the important observation is that inside the boxes $\mathcal{B}^{(\ell)}_{hor}$ we can place at least $(1-O(\varepsilon))\left|I^{(\ell)}_{hor} \cap OPT'\right| - 2|\mathcal{B}^{(\ell)}_{hor}|$ items from $I^{(\ell)}_{hor}$ as follows. Based on the slices for $I^{(\ell)}_{hor}$, we first construct a fractional packing of $\frac{1}{1+O(\varepsilon)}\text{opt}^{(\ell)}_{hor}$ items from $I^{(\ell)}_{hor}$ in which there are at most $2|\mathcal{B}^{(\ell)}_{hor}|$ items that are fractionally assigned to a box. Then we simply drop these fractional items. We use a symmetric procedure for the sets $I^{(\ell)}_{ver}$.

▶ **Lemma 10.** *For each $\ell \in \{0, ..., \lfloor \log_{1+\varepsilon} N \rfloor\}$, in time $O_\varepsilon(nN)$ we can pack at least $(1-O(\varepsilon))\left|I^{(\ell)}_{hor} \cap OPT'\right| - 2|\mathcal{B}^{(\ell)}_{hor}|$ items from $I^{(\ell)}_{hor}$ into the boxes $\mathcal{B}^{(\ell)}_{hor}$. A symmetric statement holds for $I^{(\ell)}_{ver}$ and $\mathcal{B}^{(\ell)}_{ver}$ for each $\ell \in \{0, ..., \lfloor \log_{1+\varepsilon} N \rfloor\}$.*

Thus, we obtain a packing with $(1-O(\varepsilon))|OPT'| - 2\left(\sum_\ell |\mathcal{B}^{(\ell)}_{hor}| + |\mathcal{B}^{(\ell)}_{ver}|\right)$ items in total. Note that $\left(\sum_\ell |\mathcal{B}^{(\ell)}_{hor}| + |\mathcal{B}^{(\ell)}_{ver}|\right) \leq O_\varepsilon(\log N)$. Recall that we assumed that $|OPT'| > c_\varepsilon \log N$. Thus, by choosing c_ε sufficiently large, we can ensure that $\left(\sum_\ell |\mathcal{B}^{(\ell)}_{hor}| + |\mathcal{B}^{(\ell)}_{ver}|\right) \leq \varepsilon \cdot |OPT'|$ and hence our packing contains at least $(1-O(\varepsilon))|OPT'|$ items in total.

▶ **Lemma 11.** *For each $\varepsilon > 0$ there is a constant c_ε such that if $|OPT'| > c_\varepsilon \cdot \log N$ we can compute a solution of size $(1-\varepsilon)|OPT'|$ in time $(nN)^{O_\varepsilon(1)}$.*

5 Dynamic programming with color coding

Assume that $|OPT'| \leq c \cdot \log N$ for some given constant c (which we will later choose to be the constant c_ε defined in Section 4). We describe an algorithm that computes a solution of size $|OPT'|$ for this case in time $(nN)^{O(c)}$ such that each item of this solution is contained in a corridor in $\mathcal{C}$. Our strategy is to use color-coding [5] in order to reduce the setting of $O_\varepsilon(1)$ L- and U-corridors in $\mathcal{C}$ to the setting of only one single such corridor. Then we show how to solve this problem in polynomial time.

First, we guess $|OPT'|$. Then we color each item in I randomly with one color in $\{1, ..., |OPT'|\}$. It is easy to show that with probability at least $1/e^{|OPT'|} \geq \frac{1}{N^{O(c)}}$ all items in $|OPT'|$ have different colors, in which case we say that the coloring was *successful*. If this is the case, then for each color $d \in \{1, ..., |OPT'|\}$ we can guess in time $O_\varepsilon(1)$ which corridor in $\mathcal{C}$ contains an item of OPT' that we colored with color d. This yields $O_\varepsilon(1)^{|OPT'|} = N^{O_\varepsilon(c)}$ guesses overall. By repeating the random coloring $N^{O(c)}$ times, we can ensure that, with high probability, one of these colorings was successful. Also, we can derandomize this procedure using a k-perfect family of hash functions [5, 48], which yields the following lemma.

▶ **Lemma 12.** *In time $N^{O_\varepsilon(c)}$ we can guess a partition of $\{I_C\}_{C \in \mathcal{C}}$ of I such that for each corridor $C \in \mathcal{C}$ the set I_C contains all items from OPT' that are placed inside C.*

5.1 Routine for one corridor

Recall that we are given a corridor $C \in \mathcal{C}$ and an input set I_C of items colored with $\gamma \leq c \cdot \log N$ colors. W.l.o.g., let $\{1, \ldots, \gamma\}$ be these colors. Our goal is to place precisely one item per color inside C such that they do not overlap. Let OPT'_C denote the items of OPT' placed inside C and note that also OPT'_C contains one item of each color.

For our $(1.6+\varepsilon)$-approximation it is sufficient to consider corridors with up to three sub-corridors; however, we will next describe a procedure that works for corridors with k sub-corridors for any $k \leq 1/\varepsilon$. This extension will actually be needed to obtain a $(4/3+\varepsilon)$-approximation (see the full version).

Our strategy is to cut C recursively into pieces (see Figure 5). Whenever we make a cut, we guess the items from OPT'_C that are intersected by this cut and their placement in OPT'_C. The cut splits the considered subpart of C into two pieces and we guess the colors of the items in OPT'_C in each one of these pieces. Then, we recursively solve the subproblem defined by each piece. Guessing the colors ensures that we do not place an item twice, e.g., once in each of the two subproblems. We define our cuts such that there are only a polynomial number of possible arising pieces during the recursion, and we observe that for the guesses of the colors there are only $2^\gamma \leq N^c$ many options. Hence, we can embed this recursion into a polynomial time dynamic program.

Long chords. Formally, whenever we cut C we do this along long chords defined as follows. A *long chord* is a sequence of k axis-parallel line segments $f_1, ..., f_k$ that intuitively connect e_0 with e_{k+1}, i.e., such that for each $j \in \{1, ..., k\}$ each end-point of f_j has integral coordinates and coincides with an endpoint of f_{j-1} or f_{j+1} or lies on e_0 or e_{k+1}, see Figure 5. Note that there are two special long chords that go along the edges of C, defined by $\ell_R := e_1, ..., e_k$ and $\ell_L := e_{k+2}, ..., e_{2k+1}$.

We can compute a set $\mathcal{L}$ containing all of the $N^{O(k)}$ long chords. We fix an (unknown) partition of C into $s(C) =: k$ subcorridors $S_1, ..., S_k$ that is nice for OPT'_C. We are interested in the long chords $f_1, ..., f_k$ in $\mathcal{L}$ with the property that for each $j \in \{1, ..., k\}$ the line segment f_j is contained in S_j and it is parallel to the two parallel edges that define S_j (see Figure 5). We say that such a long chord is *consistent with* $S_1, ..., S_k$ (or just *consistent* for short). Note that we do not know $S_1, ..., S_k$ and hence we cannot determine whether a given long chord is consistent or not. However, there are two key observations

- we can subdivide C recursively along the long chords such that each arising piece is defined as the area enclosed by two given long chords and e_0 and e_{k+1} (see Figure 5),
- each consistent long chord can intersect with at most k/ε_{large} items in OPT'_C since the corridors are thin and all input items are skewed.

Subproblems of DP. Therefore, we can compute a recursive partitioning of C via a dynamic program. Each cell of the DP-table is defined by

- two long chords $\ell_1, \ell_2 \in \mathcal{L}$ that might intersect but that do not properly cross each other; together with (a part of) e_0 and e_{k+1} they define a polygon $C' \subseteq C$,
- a set of $O(k/\varepsilon_{large})$ items $I'_C \subseteq I_C$ with a non-overlapping placement of them inside C such that the interior of each item in I'_C intersects ℓ_1 or ℓ_2, and
- a set of colors $\Gamma \subseteq \{1, ..., \gamma\}$.

The subproblem encoded in this cell is to place items from I_C inside C' such that they do not overlap with the items in I'_C and such that for each color $d \in \Gamma$ we place exactly one item of color d. If this subproblem has a solution $OPT(\ell_1, \ell_2, I'_C, \Gamma)$, we store it in the corresponding DP-cell; otherwise we store *fail*.

To compute such a solution, consider any long chord $\ell \in \mathcal{L}$ that lies completely inside C' but is not identical to ℓ_1 or ℓ_2 (we would like to select a consistent long chord; however, we do not know which long chords are consistent and hence we try all of them). Let us first assume that at least one such ℓ exists. Note that ℓ divides C' into two smaller polygons C'_1, C'_2 that are surrounded by the pairs (ℓ_1, ℓ), and (ℓ, ℓ_2), respectively. Then, we consider

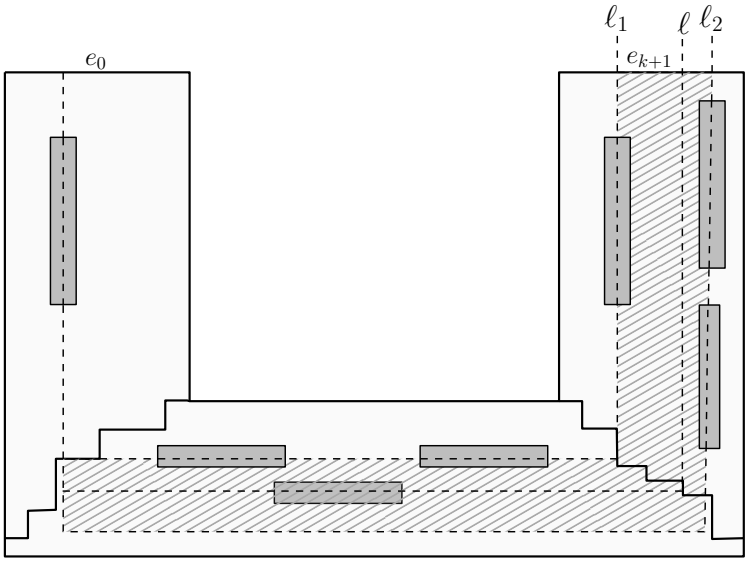


Figure 5 A U-corridor and two consistent long chords ℓ_1 and ℓ_2. The long chords intersect only a constant number of items from the optimal solution (shown in the figure). There is a DP-cell that is defined by ℓ_1, ℓ_2, and these items, and whose corresponding area is shaded. This cell splits into smaller subproblems by defining a new long chord ℓ that lies in-between ℓ_1 and ℓ_2.

any subset of items $I''_C \subseteq I_C$ and a placement of such items inside C' such that: (1) I''_C are pairwise non-overlapping and not overlapping with I'_C, (2) they are intersected by ℓ in their interior, and (3) have distinct colors $\Gamma_\ell \subseteq \Gamma$. Finally, we consider any partition $\Gamma_1 \dot\cup \Gamma_2$ of the remaining colors $\Gamma \setminus \Gamma_\ell$. Let $I'_{C,1}$ and $I'_{C,2}$ be the items in $I'_C \cup I''_C$ that intersect C'_1 and C'_2, respectively. We consider the DP-cells $(\ell_1, \ell, I'_{C,1}, \Gamma_1)$ and $(\ell, \ell_2, I'_{C,2}, \Gamma_2)$ and, if none of them contains the value "fail", we store in $(\ell_1, \ell_2, I'_C, \Gamma)$ the union of I''_C, $OPT(\ell_1, \ell, I'_{C,1}, \Gamma_1)$, and $OPT(\ell, \ell_2, I'_{C,2}, \Gamma_2)$ (together with the placement of the corresponding items) and halt the computation for the considered DP-cell. If the above event never happens, we store "fail" in this DP-cell.

The base cases of the DP are given by pairs ℓ_1, ℓ_2 which are at most one unit apart from each other (everywhere inside C), so that it is not possible to define any long chord ℓ between ℓ_1 and ℓ_2 (recall that the endpoints of the line segments of the long chords have integral coordinates). Notice however that in this case at most $O(k/\varepsilon_{large})$ skewed items can fit inside C', hence we can determine whether a feasible solution $OPT(\ell_1, \ell_2, I'_C, \Gamma)$ exists by enumeration in time $(nN)^{O(k/\varepsilon_{large})}$.

At the end we output the solution stored in the cell $(\ell_L, \ell_R, \emptyset, \{1, ..., \gamma\})$. We will show that this is the optimal solution for C. The number of DP-cells is bounded by $(nN)^{O(k/\varepsilon_{large})} \cdot 2^\gamma$ and the number of possible guesses when computing the entry of a DP-cell is bounded by $(nN)^{O(k/\varepsilon_{large})} \cdot 2^{O(\gamma)}$. This allows us to bound the running time of our DP.

▶ **Lemma 13.** *Given a path corridor C with k subcorridors and a set of skewed items I_C with γ distinct colors. In time $(nN)^{O(k/\varepsilon_{large})} \cdot 2^{O(\gamma)}$ we can determine whether there exists a set $I'_C \subseteq I_C$ with γ distinct colors that fits non-overlappingly inside C.*

We apply Lemma 13 to each corridor $C \in \mathcal{C}$ which yields the following lemma.

▶ **Lemma 14.** *Assume that $|OPT'| \leq c \cdot \log N$ for some constant c. Then we can compute a solution of size $|OPT'|$ in time $(nN)^{O(c)}$.*

Now Lemmas 11 and 14 yield the following theorem.

▶ **Theorem 15.** *There is an* $(1.6 + \varepsilon)$*-approximation algorithm with a running time of* $(nN)^{O_\varepsilon(1)}$ *for unweighted instances of 2DK with only skewed items.*

References

1 Fidaa Abed, Parinya Chalermsook, José R. Correa, Andreas Karrenbauer, Pablo Pérez-Lantero, José A. Soto, and Andreas Wiese. On guillotine cutting sequences. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2015*, volume 40 of *LIPIcs*, pages 1–19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2015. doi:10.4230/LIPIcs.APPROX-RANDOM.2015.1.

2 Anna Adamaszek, Sariel Har-Peled, and Andreas Wiese. Approximation schemes for independent set and sparse subsets of polygons. *J. ACM*, 66(4):29:1–29:40, 2019. doi:10.1145/3326122.

3 Anna Adamaszek and Andreas Wiese. Approximation schemes for maximum weight independent set of rectangles. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013*, pages 400–409. IEEE Computer Society, 2013. doi:10.1109/FOCS.2013.50.

4 Anna Adamaszek and Andreas Wiese. A quasi-ptas for the two-dimensional geometric knapsack problem. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015*, pages 1491–1505. SIAM, 2015. doi:10.1137/1.9781611973730.98.

5 Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. ACM*, 42(4):844–856, 1995. doi:10.1145/210332.210337.

6 Brenda S. Baker, Edward G. Coffman Jr., and Ronald L. Rivest. Orthogonal packings in two dimensions. *SIAM J. Comput.*, 9(4):846–855, 1980. doi:10.1137/0209064.

7 Nikhil Bansal, Alberto Caprara, Klaus Jansen, Lars Prädel, and Maxim Sviridenko. A structural lemma in 2-dimensional packing, and its implications on approximability. In *Algorithms and Computation, 20th International Symposium, ISAAC 2009*, volume 5878 of *Lecture Notes in Computer Science*, pages 77–86. Springer, 2009. doi:10.1007/978-3-642-10631-6_10.

8 Nikhil Bansal, Alberto Caprara, and Maxim Sviridenko. A new approximation method for set covering problems, with applications to multidimensional bin packing. *SIAM J. Comput.*, 39(4):1256–1278, 2009. doi:10.1137/080736831.

9 Nikhil Bansal and Arindam Khan. Improved approximation algorithm for two-dimensional bin packing. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014*, pages 13–25. SIAM, 2014. doi:10.1137/1.9781611973402.2.

10 Alberto Caprara. Packing 2-dimensional bins in harmony. In *43rd Symposium on Foundations of Computer Science (FOCS 2002)*, pages 490–499. IEEE Computer Society, 2002. doi:10.1109/SFCS.2002.1181973.

11 Parinya Chalermsook and Julia Chuzhoy. Maximum independent set of rectangles. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009*, pages 892–901. SIAM, 2009. URL: http://dl.acm.org/citation.cfm?id=1496770.1496867.

12 Timothy M. Chan and Sariel Har-Peled. Approximation algorithms for maximum independent set of pseudo-disks. *Discret. Comput. Geom.*, 48(2):373–392, 2012. doi:10.1007/s00454-012-9417-5.

13 Chandra Chekuri and Sanjeev Khanna. A polynomial time approximation scheme for the multiple knapsack problem. *SIAM J. Comput.*, 35(3):713–728, 2005. doi:10.1137/S0097539700382820.

14 Miroslav Chlebík and Janka Chlebíková. Hardness of approximation for orthogonal rectangle packing and covering problems. *J. Discrete Algorithms*, 7(3):291–305, 2009. doi:10.1016/j.jda.2009.02.002.

15 Henrik I. Christensen, Arindam Khan, Sebastian Pokutta, and Prasad Tetali. Approximation and online algorithms for multidimensional bin packing: A survey. *Comput. Sci. Rev.*, 24:63–79, 2017. doi:10.1016/j.cosrev.2016.12.001.

16 Fan Chung, Michael R. Garey, and David S. Johnson. On packing two-dimensional bins. *SIAM Journal on Algebraic Discrete Methods*, 3:66–76, 1982. doi:10.1137/0603007.

17 Julia Chuzhoy and Alina Ene. On approximating maximum independent set of rectangles. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016*, pages 820–829. IEEE Computer Society, 2016. doi:10.1109/FOCS.2016.92.

18 Edward G. Coffman Jr., M. R. Garey, David S. Johnson, and Robert Endre Tarjan. Performance bounds for level-oriented two-dimensional packing algorithms. *SIAM J. Comput.*, 9(4):808–826, 1980. doi:10.1137/0209062.

19 Florian Diedrich, Rolf Harren, Klaus Jansen, Ralf Thöle, and Henning Thomas. Approximation algorithms for 3d orthogonal knapsack. *J. Comput. Sci. Technol.*, 23(5):749–762, 2008. doi:10.1007/s11390-008-9170-7.

20 Thomas Erlebach, Klaus Jansen, and Eike Seidel. Polynomial-time approximation schemes for geometric intersection graphs. *SIAM J. Comput.*, 34(6):1302–1323, 2005. doi:10.1137/S0097539702402676.

21 Aleksei V. Fishkin, Olga Gerber, and Klaus Jansen. On efficient weighted rectangle packing with large resources. In *Algorithms and Computation, 16th International Symposium, ISAAC 2005*, volume 3827 of *Lecture Notes in Computer Science*, pages 1039–1050. Springer, 2005. doi:10.1007/11602613_103.

22 Aleksei V. Fishkin, Olga Gerber, Klaus Jansen, and Roberto Solis-Oba. Packing weighted rectangles into a square. In *Mathematical Foundations of Computer Science 2005, 30th International Symposium, MFCS 2005*, volume 3618 of *Lecture Notes in Computer Science*, pages 352–363. Springer, 2005. doi:10.1007/11549345_31.

23 Waldo Gálvez, Fabrizio Grandoni, Afrouz Jabal Ameli, Klaus Jansen, Arindam Khan, and Malin Rau. A tight $(3/2+\epsilon)$ approximation for skewed strip packing. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX-/RANDOM 2020*, volume 176 of *LIPIcs*, pages 44:1–44:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.APPROX/RANDOM.2020.44.

24 Waldo Gálvez, Fabrizio Grandoni, Sandy Heydrich, Salvatore Ingala, Arindam Khan, and Andreas Wiese. Approximating geometric knapsack via l-packings. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017*, pages 260–271. IEEE Computer Society, 2017. Full version available at http://www.dii.uchile.cl/~awiese/2DK_full_version.pdf. doi:10.1109/FOCS.2017.32.

25 Waldo Gálvez, Fabrizio Grandoni, Salvatore Ingala, and Arindam Khan. Improved pseudo-polynomial-time approximation for strip packing. In *36th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2016*, volume 65 of *LIPIcs*, pages 9:1–9:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.FSTTCS.2016.9.

26 Fabrizio Grandoni, Stefan Kratsch, and Andreas Wiese. Parameterized approximation schemes for independent set of rectangles and geometric knapsack. In *27th Annual European Symposium on Algorithms, ESA 2019*, volume 144 of *LIPIcs*, pages 53:1–53:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ESA.2019.53.

27 Rolf Harren. Approximation algorithms for orthogonal packing problems for hypercubes. *Theor. Comput. Sci.*, 410(44):4504–4532, 2009. doi:10.1016/j.tcs.2009.07.030.

28 Rolf Harren, Klaus Jansen, Lars Prädel, and Rob van Stee. A $(5/3 + \epsilon)$-approximation for strip packing. *Comput. Geom.*, 47(2):248–267, 2014. doi:10.1016/j.comgeo.2013.08.008.

29 Rolf Harren and Rob van Stee. Improved absolute approximation ratios for two-dimensional packing problems. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 12th International Workshop, APPROX 2009*, volume 5687 of *Lecture Notes in Computer Science*, pages 177–189. Springer, 2009. doi:10.1007/978-3-642-03685-9_14.

30 Sören Henning, Klaus Jansen, Malin Rau, and Lars Schmarje. Complexity and inapproximability results for parallel task scheduling and strip packing. *Theory Comput. Syst.*, 64(1):120–140, 2020. doi:10.1007/s00224-019-09910-6.

31 Sandy Heydrich and Andreas Wiese. Faster approximation schemes for the two-dimensional knapsack problem. *ACM Trans. Algorithms*, 15(4):47:1–47:28, 2019. doi:10.1145/3338512.

32 Harry B. Hunt III, Madhav V. Marathe, Venkatesh Radhakrishnan, S. S. Ravi, Daniel J. Rosenkrantz, and Richard Edwin Stearns. Nc-approximation schemes for NP- and pspace-hard problems for geometric graphs. *J. Algorithms*, 26(2):238–274, 1998. doi:10.1006/jagm.1997.0903.

33 Klaus Jansen and Malin Rau. Closing the gap for pseudo-polynomial strip packing. In *27th Annual European Symposium on Algorithms, ESA 2019*, volume 144 of *LIPIcs*, pages 62:1–62:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ESA.2019.62.

34 Klaus Jansen and Malin Rau. Improved approximation for two dimensional strip packing with polynomial bounded width. *Theor. Comput. Sci.*, 789:34–49, 2019. doi:10.1016/j.tcs.2019.04.002.

35 Klaus Jansen and Roberto Solis-Oba. New approximability results for 2-dimensional packing problems. In *Mathematical Foundations of Computer Science 2007, 32nd International Symposium, MFCS 2007*, volume 4708 of *Lecture Notes in Computer Science*, pages 103–114. Springer, 2007. doi:10.1007/978-3-540-74456-6_11.

36 Klaus Jansen and Roberto Solis-Oba. A polynomial time approximation scheme for the square packing problem. In *Integer Programming and Combinatorial Optimization, 13th International Conference, IPCO 2008*, volume 5035 of *Lecture Notes in Computer Science*, pages 184–198. Springer, 2008. doi:10.1007/978-3-540-68891-4_13.

37 Klaus Jansen and Rob van Stee. On strip packing with rotations. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, STOC 2005*, pages 755–761. ACM, 2005. doi:10.1145/1060590.1060702.

38 Klaus Jansen and Guochuan Zhang. Maximizing the number of packed rectangles. In *Algorithm Theory – SWAT 2004, 9th Scandinavian Workshop on Algorithm Theory*, volume 3111 of *Lecture Notes in Computer Science*, pages 362–371. Springer, 2004. doi:10.1007/978-3-540-27810-8_31.

39 Klaus Jansen and Guochuan Zhang. On rectangle packing: maximizing benefits. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2004*, pages 204–213. SIAM, 2004. URL: http://dl.acm.org/citation.cfm?id=982792.982822.

40 Claire Kenyon and Eric Rémila. A near-optimal solution to a two-dimensional cutting stock problem. *Math. Oper. Res.*, 25(4):645–656, 2000. doi:10.1287/moor.25.4.645.12118.

41 Arindam Khan. *Approximation algorithms for multidimensional bin packing.* PhD thesis, Georgia Institute of Technology, Atlanta, GA, USA, 2016. URL: http://hdl.handle.net/1853/54371.

42 Arindam Khan, Arnab Maiti, Amatya Sharma, and Andreas Wiese. On guillotine separable packings for the two-dimensional geometric knapsack problem. In *To appear in SoCG*, 2021.

43 Arindam Khan and Madhusudhan Reddy Pittu. On guillotine separability of squares and rectangles. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2020*, volume 176 of *LIPIcs*, pages 47:1–47:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.APPROX/RANDOM.2020.47.

44 Arindam Khan, Eklavya Sharma, and K. V. N. Sreenivas. Approximation algorithms for generalized multidimensional knapsack. *CoRR*, abs/2102.05854, 2021. arXiv:2102.05854.

45 Carla Negri Lintzmayer, Flávio Keidi Miyazawa, and Eduardo Candido Xavier. Two-dimensional knapsack for circles. In *LATIN 2018: Theoretical Informatics – 13th Latin American Symposium*, volume 10807 of *Lecture Notes in Computer Science*, pages 741–754. Springer, 2018. doi:10.1007/978-3-319-77404-6_54.

46 Arturo I. Merino and Andreas Wiese. On the two-dimensional knapsack problem for convex polygons. In *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020*, volume 168 of *LIPIcs*, pages 84:1–84:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ICALP.2020.84.

47 Giorgi Nadiradze and Andreas Wiese. On approximating strip packing with a better ratio than 3/2. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016*, pages 1491–1510. SIAM, 2016. doi:10.1137/1.9781611974331.ch102.

48 Moni Naor, Leonard J. Schulman, and Aravind Srinivasan. Splitters and near-optimal derandomization. In *36th Annual Symposium on Foundations of Computer Science, FOCS 1995*, pages 182–191. IEEE Computer Society, 1995. doi:10.1109/SFCS.1995.492475.

49 Ingo Schiermeyer. Reverse-fit: A 2-optimal algorithm for packing rectangles. In *Algorithms – ESA '94, Second Annual European Symposium*, volume 855 of *Lecture Notes in Computer Science*, pages 290–299. Springer, 1994. doi:10.1007/BFb0049416.

50 Daniel Dominic Sleator. A 2.5 times optimal algorithm for packing in two dimensions. *Inf. Process. Lett.*, 10(1):37–40, 1980. doi:10.1016/0020-0190(80)90121-0.

51 A. Steinberg. A strip-packing algorithm with absolute performance bound 2. *SIAM J. Comput.*, 26(2):401–409, 1997. doi:10.1137/S0097539793255801.

A Stepping-Up Lemma for Topological Set Systems

Xavier Goaoc
LORIA, Université de Lorraine, France

Andreas F. Holmsen
Department of Mathematical Sciences, KAIST, Daejeon, South Korea

Zuzana Patáková
Department of Algebra, Faculty of Mathematics and Physics, Charles University, Czech Republic

Abstract

Intersection patterns of convex sets in $\mathbb{R}^d$ have the remarkable property that for $d+1 \leq k \leq \ell$, in any sufficiently large family of convex sets in $\mathbb{R}^d$, if a constant fraction of the k-element subfamilies have nonempty intersection, then a constant fraction of the ℓ-element subfamilies must also have nonempty intersection. Here, we prove that a similar phenomenon holds for any topological set system $\mathcal{F}$ in $\mathbb{R}^d$. Quantitatively, our bounds depend on how complicated the intersection of ℓ elements of $\mathcal{F}$ can be, as measured by the maximum of the $\lceil \frac{d}{2} \rceil$ first Betti numbers. As an application, we improve the fractional Helly number of set systems with bounded topological complexity due to the third author, from a Ramsey number down to $d+1$. We also shed some light on a conjecture of Kalai and Meshulam on intersection patterns of sets with bounded homological VC dimension. A key ingredient in our proof is the use of the stair convexity of Bukh, Matoušek and Nivasch to recast a simplicial complex as a homological minor of a cubical complex.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Helly-type theorem, Topological combinatorics, Homological minors, Stair convexity, Cubical complexes, Homological VC dimension, Ramsey-type theorem

Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.40

Related Version *Full Version*: `https://arxiv.org/abs/2103.09286`

Funding *Xavier Goaoc*: Institut Universitaire de France.
Andreas F. Holmsen: NRF grant No. 2020R1F1A1A0104849011.
Zuzana Patáková: Charles University projects PRIMUS/21/SCI/014 and UNCE/SCI/022.

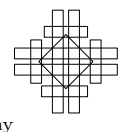
1 Introduction

In this paper, we investigate the intersection patterns of *topological set systems*, by which we mean families $\mathcal{F}$ of subsets of $\mathbb{R}^d$ such that for any $\mathcal{G} \subseteq \mathcal{F}$, the intersection of the elements of $\mathcal{G}$ has finite Betti numbers. Our main goal is to analyze how the intersection patterns of a set system $\mathcal{F}$ are influenced by the complexity of the intersection of subfamilies of $\mathcal{F}$. To measure this complexity, let us fix some integer h (set to $\lceil \frac{d}{2} \rceil$ for our purpose) and a ring to use for homology calculations ($\mathbb{Z}_2$ throughout this paper), and associate to $\mathcal{F}$ the function

$$\phi_{\mathcal{F}}^{(h)} : \begin{cases} \mathbb{N} & \to & \mathbb{N} \cup \{\infty\} \\ k & \mapsto & \sup\{\widetilde{\beta}_i(\cap\mathcal{G}) : \mathcal{G} \subseteq \mathcal{F}, |\mathcal{G}| \leq k, 0 \leq i < h\}. \end{cases}$$

We call it the (hth) *homological shatter function* of $\mathcal{F}$. Although here $\mathcal{F}$ is a set system in $\mathbb{R}^d$, the definition (and most of our methods) apply more generally (see Section 1.3).

A stepping-up phenomenon. For a set system $\mathcal{F}$, that is a family of subsets of some ground set, and an integer $k \geq 1$, let $\delta_{\mathcal{F}}(k) \in [0,1]$ denote the proportion of the k-element subsets of $\mathcal{F}$ that have nonempty intersection. Technically, our main result is the following:



▶ **Theorem 1.** *For any $d+1 \le k \le \ell$ and $b \ge 0$, for any $\delta > 0$, there exists $\delta' > 0$ such that for any sufficiently large set system $\mathcal{F}$ in $\mathbb{R}^d$, if $\phi_{\mathcal{F}}^{(\lceil \frac{d}{2} \rceil)}(\ell) \le b$ and $\delta_{\mathcal{F}}(k) \ge \delta$, then $\delta_{\mathcal{F}}(\ell) \ge \delta'$.*

Informally, Theorem 1 states that in nerves of topological set systems, positive densities tend to propagate towards *higher* dimension, in a form of "reverse Kruskal-Katona theorem". This generalizes earlier observations in combinatorial convexity and topological combinatorics, as we discuss in the next section. Moreover, the propagation rate can be bounded from below in terms of the homological shatter function.

Our result is existential and our proof does not aim at giving any reasonable estimate; the bound we obtain is an elementary recursive function, $\mathcal{E}^3$ in the Grzegorczyk hierarchy, of the parameters. Theorem 1 is, however, qualitatively sharp: without bounding the Betti numbers of intersections in *all* dimensions $0 \le j < \lceil \frac{d}{2} \rceil$ we can obtain topological set systems in $\mathbb{R}^d$ with arbitrary intersection patterns. Indeed, fix an integer $k \ge 0$ and let K be the k-skeleton of the simplex on the vertex set V. Given nonempty subsets $S_1, \ldots, S_m$ of V, define induced subcomplexes $K_1, \ldots, K_m$ where $K_i = K[S_i]$. Note that for any subset $\emptyset \ne I \subset [m]$ we have $\bigcap_{i \in I} K_i = K[\bigcap_{i \in I} S_i]$, and so any nonempty intersection of the K_i has trivial j-dimensional homology for any $j \ne k$. Since K has a geometric realization in $\mathbb{R}^d$ for any $d \ge 2k+1$, it follows that the family $\mathcal{F} = \{K_1, \ldots, K_m\}$ forms a topological set system in $\mathbb{R}^d$ where intersections have Betti numbers equal to zero except possibly in dimension k. The subsets $S_1, \ldots, S_m$ can be chosen arbitrarily, so their intersection pattern is also arbitrary.

Lowering fractional Helly numbers. One use of Theorem 1 is the reduction of *fractional Helly numbers*. For instance, it easily improves a theorem of Patáková [29, Theorem 3] into:

▶ **Corollary 2.** *For every non-negative integers b and d, there is a function $\beta_{d,b} : (0,1) \to (0,1)$ such that for any $\alpha \in (0,1)$, for any sufficiently large set system $\mathcal{F}$ in $\mathbb{R}^d$ with $\phi_{\mathcal{F}}^{(\lceil d/2 \rceil)} \le b$, if $\delta_{\mathcal{F}}(d+1) \ge \alpha$ then some $\beta_{d,b}(\alpha)|\mathcal{F}|$ members of $\mathcal{F}$ intersect.*

Specifically, [29, Theorem 3] required instead of "$\delta_{\mathcal{F}}(d+1) \ge \alpha$" that "$\delta_{\mathcal{F}}(m) \ge \alpha$" for some hypergraph Ramsey number m. The number m depends only on b and d, so we can first apply [29, Theorem 3], then follow up by Theorem 1 with $k = d+1$ and $\ell = m$. (Note that the implicit bound given by the proof of [29, Theorem 3] on the function $\beta_{d,b}$ also changes in the process.) This improvement in turn sharpens several other results, including a (p,q)-theorem, a weak ε-net theorem and a property testing algorithm. This systematic reduction of fractional Helly numbers also offers some evidence in support of some conjectures of Kalai and Meshulam [21] on a theory of *homological VC dimension*. We discuss these consequences in the next section.

1.1 Context and motivation

Let us briefly present some of the lines of research in discrete geometry, topological combinatorics and computational geometry that motivate Theorem 1.

Combinatorial convexity and beyond. The theorems of Carathéodory, Helly and Radon initiated a combinatorial theory of convexity that investigates the intersection patterns of families of convex sets, with a particular attention to the systems formed by the convex hulls of subsets of a finite point set. See the textbook of Matoušek [28, §8−10] and the surveys [8, 12, 9] for an introduction. One approach to extend combinatorial convexity is to think of a convex set as a hyperedge in an infinite hypergraph with vertex set $\mathbb{R}^d$. One can then reformulate results

from combinatorial convexity as properties of this hypergraph, and investigate whether the dependencies between theorems found in the convex setting hold for more general hypergraphs. In this sense, Alon et al. [2] established that the (reformulation of the) fractional Helly theorem implies, among others, the (reformulations of the) $(p, q)-$theorem, the weak ε-net theorem and the selection lemma, three landmarks in combinatorial convexity. (Note that this pivotal role of the fractional Helly theorem is only surpassed by Radon's lemma [17].) The study of the homological shatter functions of topological set systems recasts several previous topological relaxations of convexity [27, 24, 23, 10, 14, 29] into a broader setting.

Fractional Helly, stepping up, and upper bound theorems. The original *fractional Helly theorem* of Katchalski and Liu [26] asserts that for any $d \geq 1$ there is a function $\beta_d : (0, 1) \to (0, 1)$ such that for any $\alpha \in (0, 1)$, any finite family $\mathcal{F}$ of convex sets in $\mathbb{R}^d$ where a fraction α of the $(d + 1)$-element subsets have non-empty intersection must contain an intersecting subfamily of size at least $\beta_d(\alpha)|\mathcal{F}|$. In other words, if a positive fraction of the $(d+1)$-element subfamilies of $\mathcal{F}$ have nonempty intersection, then a positive fraction of $\mathcal{F}$ has nonempty intersection. The size of the subsets for which the "positive fraction" property is assumed, $d + 1$ here, is referred to as the *fractional Helly number*.

Katchalski and Liu [26] already observed that one can require that $\beta_d(\alpha) \to 1$ when $\alpha \to 1$. They derived it from the observation, which they dubbed the *stepping-up lemma*, that for any family $\mathcal{F}$ of convex sets in $\mathbb{R}^d$,

$$\forall d + 1 \leq k \leq \ell, \quad \delta_{\mathcal{F}}(\ell) \geq 1 - (1 - \delta_{\mathcal{F}}(k)) \binom{\ell}{k}. \tag{1}$$

In particular, if $\delta_{\mathcal{F}}(k) > 1 - 1/\binom{\ell}{k}$, then $\delta_{\mathcal{F}}(\ell) > 0$. Kalai [20] later showed that one can take $\beta_d(\alpha) \stackrel{\text{def}}{=} 1 - (1 - \alpha)^{\frac{1}{d+1}}$, which is best possible. His proof is based on the *upper bound theorem* that he [20] and Eckhoff [11] established independently. The upper bound theorem asserts that for any family $\mathcal{F}$ of n convex sets in $\mathbb{R}^d$,

$$\forall d \leq k \leq d + r - 1, \quad f_k(\mathcal{F}) > \sum_{i=0}^{d} \binom{n-r}{i} \binom{r}{k-i+1} \quad \Rightarrow \quad f_{d+r}(\mathcal{F}) > 0, \tag{2}$$

where $f_i(\mathcal{F}) = \binom{n}{i+1} \delta_{\mathcal{F}}(i + 1)$ denotes the number of $(i + 1)$-element subsets of $\mathcal{F}$ that have nonempty intersection. (That is, $f_i(\mathcal{F})$ is the number of i-dimensional faces of the nerve of $\mathcal{F}$.) It was recently shown [25] that the propagation phenomenon revealed by Equations (1) and (2) also holds for set systems in $\mathbb{R}^2$ (or on a surface) with bounded 1st homological shatter function, indicating that this phenomenon extends far beyond convexity. Our Theorem 1 gives further evidence of this by generalizing [25, Theorem 2.2] to arbitrary dimension.

Collapsibility, Leray number and homological VC dimension. The known proofs of the upper bound theorem (2) abstract away the geometry into some property shared by nerves of families of convex sets. The more elementary proofs apply to *d-collapsible* complexes [11, 20, 1], that is complexes that can be reduced by discrete homotopy moves (called *collapses*) to a d-dimensional complex [31, Lemma 1]. The more general proof, also due to Kalai (see Hell's PhD thesis [16, §5.2] for a presentation), applies to *d-Leray* complexes, that is complexes in which all induced subcomplexes have vanishing homology in dimension d and above.

Deeper connections between discrete geometry and topological combinatorics were suggested by Kalai and Meshulam in a program to develop a theory of *homological VC dimension*. The homological shatter function that we introduce here is already apparent in that program. Two of their conjectures, when combined, suggest that topological set systems with polynomial homological shatter function should enjoy a fractional Helly theorem:

▶ **Conjecture 3** (Following Kalai and Meshulam [21, Conjectures 6 and 7]). *For any integer $0 \leq k \leq d$ and any $A > 0$, there exists a function $\beta : (0,1) \to (0,1)$ such that the following holds. For any $\alpha > 0$ and any sufficiently large set system $\mathcal{F}$ in $\mathbb{R}^d$ such that $\forall m \geq 0$, $\phi_{\mathcal{F}}^{(d)}(m) \leq Am^k$, if $\delta_{\mathcal{F}}(d+1) \geq \alpha$ then some $\beta(\alpha)|\mathcal{F}|$ members of $\mathcal{F}$ have a point in common.*

This combination of Conjectures 6 and 7 from [21], also appeared in [22, Conjecture 17], except that we took upon ourselves to dissociate the dimension d of the space and the order k of the homological shatter function. Our Corollary 2 settles the case $k = 0$ of this conjecture. Moreover, our Theorem 1 reveals that for $k \geq 1$, if the assumptions of Conjecture 3 imply any fractional Helly theorem at all, then the fractional Helly number can be brought down to $d + 1$. (Note that Corollary 2 and Theorem 1 need only control $\phi_{\mathcal{F}}^{(\lceil \frac{d}{2} \rceil)}$, not $\phi_{\mathcal{F}}^{(d)}$.)

Property testing. The fact that fractional Helly theorems ensure the existence of small weak ε-nets already give them potential for computational applications. Let us stress another connection between fractional Helly theorems and algorithms, which comes from the area of *property testing*. Recall that Helly's theorem relates to the size of witness sets for convex programming, so that its generalizations, the so-called *Helly-type theorems*, correspond to the combinatorial dimension of *LP-type problems* [3] (see also [14, §1.3]). Similarly, generalizations of the fractional Helly theorem lead to *property testing* algorithms for optimization under constraints, by relating the probability that a random choice of k constraints is satisfiable to the size of the largest subset of constraints that can be simultaneously satisfied. (Here, k denotes the fractional Helly number.) This relation was spelled out by Chakraborty et al. [7] in the convex settings and holds more generally. Again, notice that reducing a fractional Helly number also improves, in principle, the efficiency of the related property testing algorithm.

1.2 Approach and further results

At a high-level, we prove Theorem 1 by a two-stage approach that we learned from Bárány and Matoušek [5], who use it for convex lattice sets. We first identify (or, in our case, prove) a Helly-type theorem that turns some intersection pattern on the k-element subsets of some family of *constant size* into *at least one* intersection of $k + 1$ sets. We then use the positive density assumption $\delta_{\mathcal{F}}(k) > 0$ to find many occasions to apply that constant-size theorem.

Supersaturation brings out colors... For this approach to work, we need the intersection pattern used in the first stage to be "massively unavoidable" when $\delta_{\mathcal{F}}(k) > 0$. Here, some extremal hypergraph theory comes into play, as in Matoušek's words [28, §9.2], "*Hypergraphs with many edges need not contain complete hypergraphs, but they have to contain complete multipartite hypergraphs*". So, our Helly theorem for constant size should not rely on a complete intersection pattern, as in Helly's original theorem, but rather on a complete k-partite intersection pattern, as in the *colorful Helly theorem* [4]. The fact that complete k-partite intersection patterns can be found in abundance as soon as $\delta_{\mathcal{F}}(k) > 0$ follows from the *supersaturation* theorem of Erdős and Simonovits; We postpone its presentation to Section 7, as we will only need it (and the related terminology) in the final step of our proof.

... and colors lead to stair convexity. Given set systems $\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_m$, a subfamily $\mathcal{G} \subset \bigcup_{i=1}^m \mathcal{F}_i$ is called *colorful* (with respect to the families $\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_m$) if $\mathcal{G}$ contains at most one member from each $\mathcal{F}_i$. Here is our colorful Helly theorem:

▶ **Theorem 4.** *For any integers $b \geq 0$ and $m > d \geq 1$ there exists an integer $t = t(b, d, m)$ such that for any topological set systems $\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_m$ in $\mathbb{R}^d$, each of size t, if every colorful subfamily $\mathcal{G}$ satisfies $\phi_{\mathcal{G}}^{(\lceil d/2 \rceil)} \leq b$ and has nonempty intersection, then some $2m - d$ members of $\bigcup_{i=1}^m \mathcal{F}_i$ have nonempty intersection.*

We prove Theorem 4 using a technique developed in [14, 29] for studying intersection patterns via homological minors [30]. In short, a cellular chain complex C_1 is a minor of a cellular chain complex C_2 if there is a non-trivial chain map $C_1 \to C_2$ that sends disjoint faces to chains with disjoint supports. One novelty here is that, in order to account for the k-partite structure of the color classes, we work with minors in chain complexes built out of cubical cells rather than simplices. For this, we have to transfer some results from simplicial homology, like the homological Van Kampen-Flores Theorem (adapted in Proposition 8). We show that the stair convexity of Bukh et al. [6] offers a systematic way of building chain maps from simplicial complexes into grid-like complexes (Proposition 7). Along the way, we also establish a new Ramsey-type result (Lemma 9) which we use to construct grid-like minors inside the intersections of sets with bounded homological shatter function.

Further consequences. Let us say a word on some of the consequences of the fractional Helly theorem as spelled out by Alon et al. [2]. Our Corollary 2, via Theorems 8(i) and 9 and the discussion in §2.1 in [2], implies:[1]

▶ **Corollary 5.** *For every d, b and $p \geq d+1$, there exists $\tau = \tau(d, b, p)$ such that the following holds. Let $\mathcal{F}$ be a set system in $\mathbb{R}^d$ with $\phi_{\mathcal{F}}^{(\lceil d/2 \rceil)} \leq b$. If among any p members of $\mathcal{F}$, some $d+1$ intersect, then there exists a set of τ points of $\mathbb{R}^d$ that intersects every member of $\mathcal{F}$.*

Here, without our sharpening of Corollary 2 by Theorem 1, it would take any $p \geq m$ members of $\mathcal{F}$ to contain some m intersecting members, for some hypergraph Ramsey number m. Similarly, our Corollary 2 together with the Theorem 9 and the discussion in §2.1 from [2] imply that for any b and d, there are c_1 and c_2 such that any topological set system in $\mathbb{R}^d$ with $\lceil d/2 \rceil$th homological shatter function bounded from above by b admits a weak ε-net of size c_1/ε^{c_2}. Here, the effect of our sharpening is to reduce the constants c_1 and c_2.

1.3 Remarks and open questions

1. Do nerves of topological set systems with finite homological shatter functions have bounded Leray number? Indeed Theorem 1 reveals that these nerves enjoy some of the consequences of d-Lerayness. A first step in this direction was done by Holmsen, Kim and Lee [18], but the question remains open, already for a homological shatter function bounded by a *constant*.
2. In the proof of Theorem 1, the assumption that the sets are in $\mathbb{R}^d$ is used in exactly one place, namely in Section 4 when we invoke the homological version of the Van Kampen-Flores theorem [14, Corollary 14]. The proof therefore generalizes readily to, for instance, topological set systems in a manifold with some forbidden homological minor.

[1] More precisely, the statements follow from Corollary 2 applied to the family $\mathcal{F}^{\cap} \stackrel{\text{def}}{=} \{\cap_{S \in \mathcal{G}} S \colon \mathcal{G} \subset \mathcal{F}\}$. Note that $\phi_{\mathcal{F}^{\cap}}^{(\lceil d/2 \rceil)}$ is bounded from above by b if and only if $\phi_{\mathcal{F}}^{(\lceil d/2 \rceil)}$ is.

3. The idea behind stretched grids [6] suggest a general transfer principle between stair convex hulls and (limits of) simplices in $\mathbb{R}^d$. Perhaps this could offer a more conceptual approach to proving Proposition 7. Our efforts in that direction were not fruitful.

2 Background

We write $\mathbb{N} = \{1, 2, \ldots\}$ for the set of positive integers and $\mathbb{N}_0 = \{0, 1, \ldots\}$ for the set of non-negative integers. We write $[n] \stackrel{\text{def}}{=} \{1, 2, \ldots, n\}$ and $\binom{[n]}{k}$ for the set of k-element subsets of $[n]$. All homology in this paper is computed with coefficients in $\mathbb{Z}_2$, and we write $C_\#\left(\mathbb{R}^d\right)$ for the singular chain complex of $\mathbb{R}^d$ with $\mathbb{Z}_2$ coefficients. Given a chain α in a chain complex, we write $\operatorname{supp}(\alpha)$ for its support. We use homology on a family of cubical complexes, in the sense, *e.g.*, of Kaczynski et al. [19], which we now define.

2.1 Grid complexes

Let $G[n]$ denote the 1-dimensional cell complex whose vertices (0-cells) are the singletons $\{1\}, \{2\}, \ldots, \{n\}$ and whose closed 1-cells are the closed intervals $[1, 2], [2, 3], \ldots, [n-1, n]$. For $m \geq 1$, define the *grid complex* $G[n]^m$ as the m-fold product $G[n]^m \stackrel{\text{def}}{=} \underbrace{G[n] \times \cdots \times G[n]}_{m\text{-fold}}$, equipped with the product topology.

Cells. For every integer $a \in [n]$ we use interchangeably the notations $[a, a] = \{a\}$ to denote the corresponding 0-cell in $G[n]$. For every integers $a, b \in [n]$ with $a < b$ we let $[a, b] = [b, a]$ denote the 1-chain with $\mathbb{Z}_2$ coefficients

$$\left.\begin{matrix}[a,b]\\[b,a]\end{matrix}\right\} \stackrel{\text{def}}{=} [a, a+1] + [a+1, a+2] + \cdots + [b-1, b].$$

For any *pairwise distinct* integers $a, b, c \in [n]$ we have $[a, c] = [a, b] + [b, c]$. Every (closed) k-cell σ in $G[n]^m$ can be written as the product of exactly $(m-k)$ 0-cells and k 1-cells

$$\sigma = [a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_m, b_m],$$

where $1 \leq a_i \leq b_i \leq a_i + 1 \leq n$. A *$k$-chain* is a sum of k-cells in $G[n]^m$ with coefficients in $\mathbb{Z}_2$. We note that $G[n]^m$ is a regular cell complex of dimension m which can be realized geometrically as an m-dimensional axis-parallel cube in $\mathbb{R}^m$ with sidelength $n-1$. For $\ell \leq m$, the set of cells of dimension at most ℓ of $G[n]^m$ is a regular cell complex of dimension ℓ, called the *ℓ-dimensional skeleton* of $G[n]^m$ and denoted by $(G[n]^m)^{(\ell)}$. For X a subcomplex of a grid complex, we write $V(X)$ for the set of vertices of X.

Products and boundaries. The *product* $\times$ of a k_1-cell of $G[n]^{m_1}$ by a k_2-cell of $G[n]^{m_2}$ is a $(k_1 + k_2)$-cell of $G[n]^{m_1+m_2}$. We extend it to chains by putting

$$(\sigma_1 + \cdots + \sigma_{\ell_1}) \times (\tau_1 + \cdots + \tau_{\ell_2}) \stackrel{\text{def}}{=} \sum_{i=1}^{\ell_1} \sum_{j=1}^{\ell_2} \sigma_i \times \tau_j.$$

We denote the null chain (with empty support) by 0 and clarify that for any chain σ we have $\sigma \times 0 = 0 \times \sigma = 0$. We can now define the *boundary* of a cell of $G[n]^m$ recursively, as follows:

$$\begin{array}{rrcll}
\textbf{(0-cells)} & \partial\{a\} & \stackrel{\text{def}}{=} & 0 & \text{(the trivial chain)} \\
\textbf{(1-cells)} & \partial[a, a+1] & \stackrel{\text{def}}{=} & \{a\} + \{a+1\} & \\
\textbf{(}\geq \textbf{2-cells)} & \partial(\sigma \times \tau) & \stackrel{\text{def}}{=} & \partial\sigma \times \tau + \sigma \times \partial\tau &
\end{array}$$

The definition of ∂ extends from k-cells to k-chains by linearity. For X a (skeleton of a) grid complex, we write $C_{\#}(X)$ for the chain complex defined by the chains of X together with ∂.

2.2 Stair convexity

The stair convex hull was introduced by Bukh, Matoušek, and Nivasch [6] as a tool for analyzing point configurations and extremal problems in discrete geometry such as lower bounds on the size of weak ε-nets. We now reformulate this notion in terms of chains of the grid complex $G[n]^m$; this resembles their recursive definition.

Stair convex chains. We fix some integer $n \geq 2$ and work, implicitly, in the grid complexes $G[n]^m$. For any $m \geq k \geq 0$ and any integers $1 \leq a_1 < \cdots < a_{k+1} \leq n$ we define a *stair convex* k-chain $\mathrm{sc}^m(a_1, \ldots, a_{k+1}) \in C_k(G[n]^m)$, which we also denote by $\mathrm{sc}^m_k(a_1, \ldots, a_{k+1})$ when we want to make its dimension explicit. The definition is recursive:

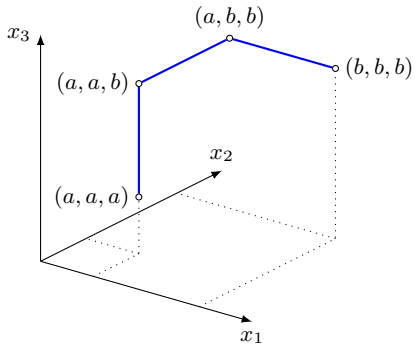
$$\begin{array}{lrcl}
(\mathbf{k = 0}) & \mathrm{sc}^m(a) & \stackrel{\text{def}}{=} & \overbrace{(a, \ldots, a)}^{m\text{-fold}} \\
(\mathbf{k > m}) & \mathrm{sc}^m(a_1, \ldots, a_{k+1}) & \stackrel{\text{def}}{=} & 0 \qquad \text{(the trivial chain)} \\
(\mathbf{0 < k < m}) & \mathrm{sc}^m(a_1, \ldots, a_{k+1}) & \stackrel{\text{def}}{=} & \mathrm{sc}^{m-1}_{k-1}(a_1, \ldots, a_k) \times [a_k, a_{k+1}] \\
 & & & \quad + \ \mathrm{sc}^{m-1}_k(a_1, \ldots, a_{k+1}) \times \{a_{k+1}\} \\
(\mathbf{0 < k = m}) & \mathrm{sc}^m(a_1, \ldots, a_{m+1}) & \stackrel{\text{def}}{=} & [a_1, a_2] \times [a_2, a_3] \times \ldots \times [a_m, a_{m+1}]
\end{array}$$

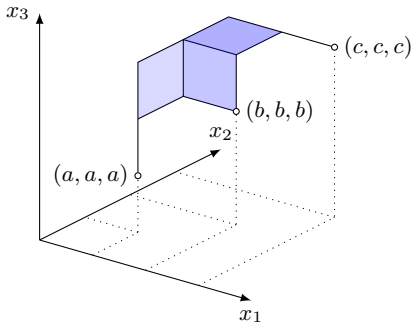
First examples. At one end, for $k = 0$, $\mathrm{sc}^m(a)$ is a vertex on the diagonal of $G[n]^m$. At the other end, for $k = m$, $\mathrm{sc}^m(a_1, \ldots, a_{m+1})$ is a m-dimensional box. Let us examine some simple examples of what happens in-between. For $m = 2$ and $k = 1$ we have

$$\mathrm{sc}^2(a, b) = \ \mathrm{sc}^1(a) \times [a, b] + \mathrm{sc}^1(a, b) \times \{b\} = \{a\} \times [a, b] + [a, b] \times \{b\}$$

which is a rectilinear path from (a, a) to (b, b) with a bend at (a, b). Here are more examples:

$$\begin{aligned}
&\mathrm{sc}^3(a, b) \\
&= \mathrm{sc}^2(a) \times [a, b] + \mathrm{sc}^2(a, b) \times \{b\} \\
&= (a, a) \times [a, b] + (\{a\} \times [a, b] + [a, b] \times \{b\}) \times \{b\} \\
&= (a, a) \times [a, b] + \{a\} \times [a, b] \times \{b\} + [a, b] \times (b, b)
\end{aligned}$$





$$\begin{aligned}&\mathrm{sc}^3(a,b,c)\\&=\mathrm{sc}^2(a,b)\times[b,c]+\mathrm{sc}^2(a,b,c)\times\{c\}\\&=\{a\}\times[a,b]\times[b,c]\\&\quad+[a,b]\times\{b\}\times[b,c]\\&\quad\quad+[a,b]\times[b,c]\times\{c\},\end{aligned}$$

A non-recursive definition. Let us extend the definition of stair convex hull to the case $m=k=0$ by putting, for any integer a and any chain σ, $\mathrm{sc}_0^0(a)\times\sigma=\sigma\times\mathrm{sc}_0^0(a)=\sigma$. With this convention, we can "unwrap" the recursive definition of sc_k^m:

$$\mathrm{sc}_k^m(a_1,\dots,a_{k+1})=\sum_{\substack{t_1,t_2,\dots,t_{k+1}\in\mathbb{N}_0\\ t_1+t_2+\dots+t_{k+1}=m-k}}\mathrm{sc}_0^{t_1}(a_1)\times\prod_{i=1}^{k}\Big([a_i,a_{i+1}]\times\mathrm{sc}_0^{t_{i+1}}(a_{i+1})\Big)\tag{3}$$

So, for instance, $\mathrm{sc}^3(a,b)=\{a\}^2\times[a,b]+\{a\}\times[a,b]\times\{b\}+[a,b]\times\{b\}^2$. From (3) we get:

▶ **Lemma 6.** *If an axis-aligned hyperplane $x_j=a$ contains a k-dimensional face of the support of $\mathrm{sc}_k^m(a_1,\dots,a_{k+1})$, then $a\in\{a_1,\dots,a_{k+1}\}$.*

Proof. The k-dimensional faces of the support of $\mathrm{sc}_k^m(a_1,\dots,a_{k+1})$ are the summands of the right hand term of (3). A summand is contained in $x_j=a$ only if for some i we have $t_i\neq 0$ and $a_i=a$. ◀

3 Stair convex hulls and boundary operator

The key property of stair convex hull for our purpose is that under some conditions, it behaves like k-dimensional simplices with respect to the boundary operator on grid complexes. Figure 1 illustrates this phenomenon in 2 dimensions. To formalize this claim, let us write $(a_1,\dots,\widehat{a_i},\dots,a_{k+1})\stackrel{\text{def}}{=}(a_1,\dots,a_{i-1},a_{i+1},\dots,a_{k+1})$, that is, $\widehat{\ }$ denotes coordinates to be omitted. (Recall that all homology in this paper has coefficients in $\mathbb{Z}_2$.)

▶ **Proposition 7.** *For integers $m\geq k\geq 1$ and any sequence $a_1<a_2<\ldots<a_m$ of elements from $[n]$ we have $\partial\,\mathrm{sc}_k^m(a_1,\dots,a_{k+1})=\sum_{i=1}^{k+1}\mathrm{sc}_{k-1}^m(a_1,\dots,\widehat{a_i},\dots,a_{k+1})$.*

Sketch of proof. Our proof is a (not so short) calculation. We set up an induction on m by using the recursive definition of sc_k^m (for $k<m$) or by applying the product rule after singling out the factor $[a_m,a_{m+1}]$ (for $k=m$). One important idea is to handle the factors $[a_{i-1},a_{i+1}]$ arising from $\mathrm{sc}_{k-1}^m(a_1,\dots,\widehat{a_i},\dots,a_{k+1})$ using the identity $[a_{i-1},a_{i+1}]=[a_{i-1},a_i]+[a_i,a_{i+1}]$ between 1-chains. See the full version for the details. ◀

4 A homological van Kampen-Flores theorem for grid complexes

We now use Proposition 7 to prove a non-embeddability result, in homological terms, that is well-suited for grid complexes. Following [30], we call chain map *non-trivial* if the image of every vertex is a 0-chain supported on an odd number of vertices.

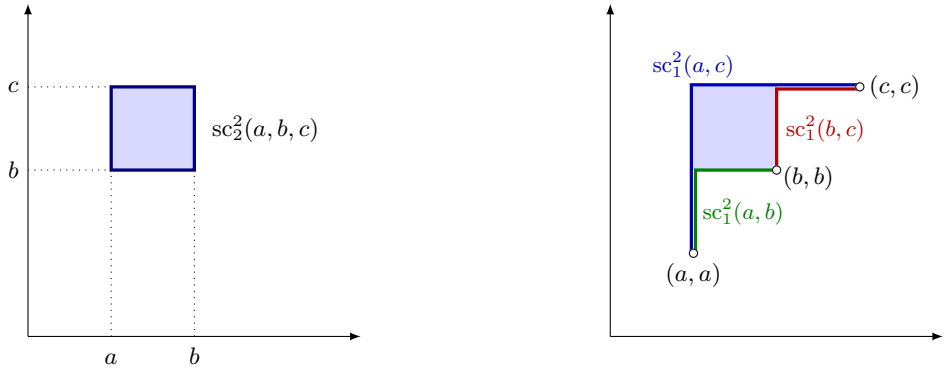


Figure 1 Left: An illustration of the 2-chain $\mathrm{sc}_2^2(a,b,c)$ with highlighted boundary. Right: An illustration of the boundary of $\mathrm{sc}_2^2(a,b,c)$ decomposed into the sum of $\mathrm{sc}_1^2(a,b), \mathrm{sc}_1^2(b,c)$ and $\mathrm{sc}_1^2(a,c)$, respectively. Note that both $\{a\}\times[a,b]$ and $[b,c]\times\{c\}$ cancel out since we work with $\mathbb{Z}_2$ coefficients.

▶ **Proposition 8.** *Let $m > d \geq 1$ be integers and let X be the $\lceil d/2\rceil$-skeleton of the grid complex $G[d+3]^m$. For every nontrivial chain map $f_\# : C_\#(X) \to C_\#(\mathbb{R}^d)$, there exist cells σ and τ in X such that (i) $\dim\sigma + \dim\tau \leq d$, (ii) σ and τ are not contained in a common axis-parallel hyperplane, and (iii) the supports of $f_\#(\sigma)$ and $f_\#(\tau)$ intersect.*

Proof. For $p \geq 0$ even, let M_p be the $p/2$-skeleton of the $(p+2)$-dimensional simplex. For $p \geq 1$ odd, let M_p be the cone over M_{p-1}. The simplicial complex M_d satisfies:

- The dimensions of any two disjoint faces in M_d sum to at most d. Indeed, for d odd, every face of dimension $\lceil d/2\rceil$ contains the coning vertex.
- For any non-trivial chain map $C_\#(M_d) \to C_\#(\mathbb{R}^d)$, there exist two disjoint simplices of M_d whose images have non-disjoint supports. In other words, M_d enjoys a homological van Kampen-Flores theorem. For d even, it follows from a standard proof of the van Kampen–Flores theorem through the *Van Kampen obstruction* [14, § 2]. The case where d is odd can be reduced to the even case $d-1$ using properties of this obstruction with respect to coning, see the proof of [14, Corollary 14].

Let us label the vertices of M_d by $v_1, v_2, \ldots, v_{d+3}$. For every simplex $\{v_{i_1}, \ldots, v_{i_{k+1}}\}$ in M_d with $i_1 < i_2 < \ldots < i_{k+1}$, we let $g(\{v_{i_1}, \ldots, v_{i_{k+1}}\}) \stackrel{\text{def}}{=} \mathrm{sc}^m(i_1, \ldots, i_{k+1})$. We extend g linearly into a map $g_\# : C_\#(M_d) \to C_\#(X)$ and note that $g_\#$ is a chain map, as Proposition 7 ensures that for any simplex $\sigma \in C_\#(M_d)$ we have $\partial(g_\#(\sigma)) = g_\#(\partial\sigma)$. Now, let us consider the chain map $f_\# \circ g_\# : M_d \to \mathbb{R}^d$. It is non-trivial, so by the homological van Kampen–Flores theorem there exist two disjoint simplices σ_M and τ_M of M_d whose images under $f_\# \circ g_\#$ have non-disjoint supports. So, there exists a cell σ in the support of $g(\sigma_M)$ with $\dim\sigma \leq \dim\sigma_M$ and a cell τ in the support of $g(\tau_M)$ with $\dim\tau \leq \dim\tau_M$ such that $f_\#(\sigma)$ and $f_\#(\tau)$ have non-disjoint support. Note that σ and τ satisfy condition (iii). The dimensions of σ and τ, like the dimensions of σ_M and τ_M, sum to at most d, so condition (i) is also satisfied. Finally, since σ_M an τ_M are disjoint, Lemma 6 ensures that σ and τ are not contained in a common axis-aligned hyperplane, and condition (ii) is satisfied as well. ◀

Note that a k-dimensional cell of $G[n]^m$ has $m-k$ coordinates that are constant. So, for any two simplices σ_1, σ_2 of X such that $\dim\sigma_1 + \dim\sigma_2 < m$, there is at least one coordinate that is constant for both. In particular, if two such simplices intersect they must be contained

in a common axis-parallel hyperplane. So Conditions (i) and (ii) imply that σ and τ are vertex-disjoint, as in the usual van Kampen–Flores theorem. More generally, the chain map $g_\#$ maps disjoint simplices to chains with disjoint support: in the general sense of [30], it is a homological almost embedding that shows that M_d is a homological minor of X.

We have no reason to believe that the complex $X = (G[d+3]^m)^{(\lceil d/2 \rceil)}$ is a minimal one for which the conclusion of Proposition 8 holds. For instance, it is easy to see that the conclusion holds for chain maps $C_\#((G[2]^m)^{(1)}) \to C_\#(\mathbb{R}^1)$. Furthermore, we could show that the conclusion also holds for chain maps $C_\#((G[3]^3)^{(1)}) \to C_\#(\mathbb{R}^2)$ by computation of the van Kampen obstruction (but were unable to extend this to a general approach).

5 Filling holes: A Ramsey-type result for grid complexes

We now prove the last ingredient of our colorful Helly type Theorem 4: a Ramsey-type result.

5.1 Subgrids and the subgrid lemma

The structure that our Ramsey-type result identifies is a subgrid of $G[n]^m$. Formally, a *subgrid* of $G[n]^m$ is a map $\gamma : V(G[\ell]^m) \to V(G[n]^m)$, for some $1 \le \ell \le n$, given by $(x_1, \dots, x_m) \longmapsto (\gamma_1(x_1), \dots, \gamma_m(x_m))$, where each $\gamma_i : [\ell] \to [n]$ is a strictly increasing function. We write the fact that γ is a subgrid by $\gamma : G[\ell]^m \hookrightarrow G[n]^m$. For any $a, b \in [n]$ we let $\gamma_i(\{a\}) \stackrel{\text{def}}{=} \{\gamma_i(a)\}$ and $\gamma_i([a,b]) \stackrel{\text{def}}{=} [\gamma_i(a), \gamma_i(b)]$, and let

$$\gamma_\# : \begin{cases} C_\#(G[\ell]^m) & \to & C_\#(G[n]^m) \\ \sigma_1 \times \cdots \times \sigma_m & \mapsto & \gamma_1(\sigma_1) \times \cdots \times \gamma_m(\sigma_m). \end{cases}$$

As we spell out in the full version, $\gamma_\#$ is a chain map. Now, fix an integer $k \ge 1$ and consider a group homomorphism h from the group $(C_k(G[n]^m), +)$ of k-chains into $(\mathbb{Z}_2)^b$, where $b \in \mathbb{N}$. We say that a subgrid $\gamma : G[a]^m \hookrightarrow G[n]^m$ *lies in the kernel of* h if $h(\gamma_\#(c)) = 0$ for every $c \in C_k(G[a]^m)$. Here is our Ramsey-type statement:

▶ **Lemma 9** (subgrid lemma). *For any $b, k, m, \ell \in \mathbb{N}$, $\ell \ge 2$, there exists $N = N(b,k,m,\ell)$ such that for any group homomorphism $h : C_k(G[N]^m) \to (\mathbb{Z}_2)^b$, there exists a subgrid $\gamma : G[\ell]^m \hookrightarrow G[N]^m$ in the kernel of h.*

5.2 Proof of the subgrid lemma

The rest of this section is devoted to the proof of Lemma 9. It may help the reader to first read the rest of this section once with the sole case $k = m$ in mind.

Coloring vertices. We first establish a simple Ramsey-type property of subgrids:

▶ **Lemma 10.** *For any m, ℓ and q there exists $N = N(m,\ell,q)$ such that for every q-coloring of $V(G[N]^m)$, there exists a monochromatic subgrid $G[\ell]^m \hookrightarrow G[N]^m$.*

This follows for instance from the Gallai-Witt theorem [15, p. 40], but it is in fact much simpler as it dispenses of the "homothetic" constraint. See the full version for a direct proof.

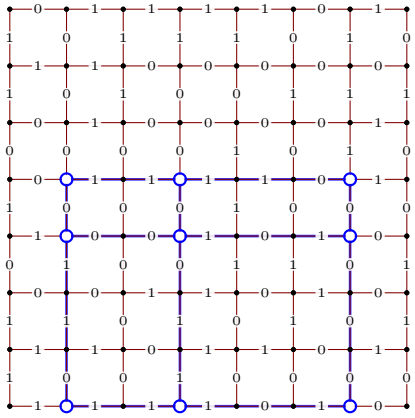


Figure 2 A homomorphism $h : C_1(G[8]^2) \to \mathbb{Z}_2$. The blue subgrid $\gamma : G[3]^2 \hookrightarrow G[8]^2$ lies in the kernel of h.

Boxes. Let $\mathbf{1} \stackrel{\text{def}}{=} (1, 1, \ldots, 1) \in [N]^m$. Given two grid points $x, y \in [N]^m$ we write $x \preceq y$ if $x_i \leq y_i$ for $1 \leq i \leq m$. We also put $\operatorname{diff}(x, y) \stackrel{\text{def}}{=} \{i \in [m] : x_i \neq y_i\}$. For any two vertices $x \preceq y$ in $G[N]^m$ we put

$$\operatorname{box}_k(x, y) \stackrel{\text{def}}{=} \begin{cases} 0 & \text{if } |\operatorname{diff}(x, y)| \neq k, \\ [x_1, y_1] \times [x_2, y_2] \times \cdots \times [x_m, y_m] & \text{otherwise.} \end{cases}$$

That is, $\operatorname{box}_k(x, y)$ is a k-chain. It is non-trivial if and only if x and y disagree on exactly k coordinates. In that case, it is a k-dimensional box, contained in the intersection of all coordinate hyperplanes where x and y agree.

Shuffles. For any $I \subset [m]$ and $x, y \in [N]^m$ we define the *I-shuffle* of x and y as

$$\lfloor x, y \rceil_I \stackrel{\text{def}}{=} (z_1, z_2, \ldots, z_m) \quad \text{where} \quad z_i = \begin{cases} x_i & \text{if } i \in I, \text{ and} \\ y_i & \text{otherwise.} \end{cases}$$

Notice that if $\operatorname{box}_k(x, y)$ is a non-trivial k-chain, then its support is the k-dimensional box with corners $\{\lfloor x, y \rceil_I\}_{I \subset \operatorname{diff}(x,y)}$. For $z \in [\ell]^m$, we let $G_k(z)$ denote the set of vertices of $G([\ell]^m)$ that can be reached from z by incrementing exactly k coordinates, that is

$$G_k(z) \stackrel{\text{def}}{=} \{w \in [\ell]^m \colon z \preceq w, \operatorname{diff}(z, w) = k, \forall i \in [m]\ z_i \leq w_i \leq z_i + 1\}.$$

Notice that the set $\{\operatorname{box}_k(z, w)\}_{z \in [\ell]^m, w \in G_k(z)}$ generates the vector space $C_k(G[\ell]^m)$. It follows that a subgrid $\gamma : G[\ell]^m \hookrightarrow G[N]^m$ lies in the kernel of h if

$$\forall z \in [\ell]^m, \forall w \in G_k(z), \quad h\left(\operatorname{box}_k\left(\gamma(z), \gamma(w)\right)\right) = 0. \tag{4}$$

Inclusion-exclusion. We next define for any $x, y \in [N]^m$ a *base point*, also in $[N]^m$:

$$\operatorname{base}(x, y) \stackrel{\text{def}}{=} (z_1, z_2, \ldots, z_m) \quad \text{where} \quad z_i = \begin{cases} 1 & \text{if } i \in \operatorname{diff}(x, y), \\ x_i = y_i & \text{otherwise.} \end{cases}$$

Now, for any $x \preceq y$ in $[N]^m$ with $|\operatorname{diff}(x, y)| = k$, the inclusion-exclusion principle yields

$$\operatorname{box}_k(x, y) = \sum_{I \subseteq \operatorname{diff}(x,y)} (-1)^{|I|} \operatorname{box}_k\left(\operatorname{base}(x, y), \lfloor x, y \rceil_I\right). \tag{5}$$

Indeed, for $k = m$, this merely writes a full-dimensional, axis-parallel box B as the alternating sum of the boxes spanned by $\mathbf{1}$ and each of corners of B. The argument is the same for $k < m$ by simply dropping all coordinates where x and y agree. The factors $(-1)^{|I|}$, which would be necessary if the chains had coefficients in $\mathbb{Z}$, may be surprising over $\mathbb{Z}_2$. Their interest comes from considering the above identity through the homomorphism h:

$$h\left(\mathrm{box}_k(x,y)\right) = \sum_{I \subseteq \mathrm{diff}(x,y)} (-1)^{|I|} h\left(\mathrm{box}_k\left(\mathrm{base}(x,y), \lfloor x,y \rceil_I\right)\right). \tag{6}$$

Coloring. Let us associate to the group homomorphism $h : C_k(G[N]^m) \to (\mathbb{Z}_2)^b$ the coloring

$$\chi_h : \begin{cases} V(G[N]^m) & \to & (\mathbb{Z}_2)^{b\binom{m}{k}} \\ z & \mapsto & \left(h\left(\mathrm{box}_k\left(\lfloor \mathbf{1}, z \rceil_F, z\right)\right)\right)_{F \in \binom{[m]}{k}} \end{cases}$$

In plain english, $\chi_h(z)$ is a vector of $\binom{m}{k}$ elements of $(\mathbb{Z}_2)^b$, one per k-element subset $F \subseteq [m]$ (the order in which these subsets are considered is irrelevant). The element of $(\mathbb{Z}_2)^b$ associated to subset F is obtained by considering the k-dimensional axis parallel subspace through x where coordinates with index in F are fixed: $(\chi_h(z))_F$ is the image under h of the k-dimensional box spanned by the base point $\lfloor \mathbf{1}, z \rceil_F$ and z.

Wrapping up. Let $N_0 \stackrel{\text{def}}{=} N\left(m, \ell, 2^{b\binom{m}{k+1}}\right)$ for the function $N(\cdot,\cdot,\cdot)$ from Lemma 10. So, if $N \geq N_0$, then there exists a subgrid $\gamma : G[\ell]^m \hookrightarrow G[N]^m$ that is monochromatic for χ_h.

To argue that γ lies in the kernel of h, we use Condition (4): we consider some arbitrary $z \in [\ell]^m$ and $w \in G_k(z)$, let $x \stackrel{\text{def}}{=} \gamma(z)$ and $y \stackrel{\text{def}}{=} \gamma(w)$, and set out to prove that $h(\mathrm{box}_k(x,y)) = 0$. Note that the fact that γ is a subgrid and $w \in G_k(z)$ ensures that $|\mathrm{diff}(x,y)| = k$. For $I \subseteq \mathrm{diff}(x,y)$ let us write $c_I \stackrel{\text{def}}{=} \lfloor x,y \rceil_I$. Each coordinate of c_I comes from either $x = \gamma(z)$ or $y = \gamma(w)$, so c_I is a vertex of the subgrid γ. Moreover, for every $i \notin \mathrm{diff}(x,y)$ we have $x_i = y_i = (c_I)_i$ so $\mathrm{base}(x,y) = \lfloor \mathbf{1}, x \rceil_{\mathrm{diff}(x,y)} = \lfloor \mathbf{1}, y \rceil_{\mathrm{diff}(x,y)} = \lfloor \mathbf{1}, c_I \rceil_{\mathrm{diff}(x,y)}$. Equation (6) then rewrites as

$$h\left(\mathrm{box}_k(x,y)\right) = \sum_{I \subseteq \mathrm{diff}(x,y)} (-1)^{|I|} h\left(\mathrm{box}_k\left(\lfloor \mathbf{1}, c_I \rceil_{\mathrm{diff}(x,y)}, c_I\right)\right). \tag{7}$$

Notice that the value $h\left(\mathrm{box}_k\left(\lfloor \mathbf{1}, c_I \rceil_{\mathrm{diff}(x,y)}, c_I\right)\right)$ is independent of $I \subseteq \mathrm{diff}(x,y)$. This follows from the facts that (i) $|\mathrm{diff}(x,y)| = k$, (ii) c_I is a vertex of the subgrid γ, and (iii) γ is monochromatic for χ_h. Equation (7) therefore rewrites as

$$h\left(\mathrm{box}_k(x,y)\right) = h\left(\mathrm{box}_k\left(\lfloor \mathbf{1}, c_\emptyset \rceil_{\mathrm{diff}(x,y)}, c_\emptyset\right)\right) \left(\sum_{I \subseteq \mathrm{diff}(x,y)} (-1)^{|I|}\right) = 0.$$

This concludes the proof of the subgrid lemma.

6 A weak colorful Helly theorem

We now set out to prove Theorem 4. Recall that we are given arbitrary integers $b \geq 0$ and $m > d \geq 1$. Our task is to prove that there exists an integer $t = t(b,d,m)$ such that for any topological set systems $\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_m$ in $\mathbb{R}^d$, each of size t, if every colorful subfamily $\mathcal{G}$ satisfies $\phi_{\mathcal{G}}^{(\lceil d/2 \rceil)} \leq b$ and the members of $\mathcal{G}$ have nonempty intersection, then some $2m - d$ members of $\bigcup_{i=1}^m \mathcal{F}_i$ have nonempty intersection. Before we state the main technical step of our proof, Lemma 14, we need some definitions.

6.1 Setup

We define constants $t_0 > t_1 > \cdots > t_{\lceil d/2 \rceil}$ starting with $t_{\lceil d/2 \rceil} \stackrel{\text{def}}{=} d+3$ and, having defined t_{i+1}, setting $t_i \stackrel{\text{def}}{=} N(b, i, m, t_{i+1})$ where $N(\cdot,\cdot,\cdot,\cdot)$ is the function from the subgrid lemma (Lemma 9). We prove Theorem 4 for $t(b,d,m) \stackrel{\text{def}}{=} t_0$. We let $X_i \stackrel{\text{def}}{=} G[t_i]^m$ and note that the definition of the t_i's ensures:

▷ Claim 11. For any homomorphism $h : C_i(X_i) \to (\mathbb{Z}_2)^b$, there exists a subgrid $\gamma : X_{i+1} \hookrightarrow X_i$ in the kernel of h.

We label the members of each $\mathcal{F}_i$ arbitrarily as $\mathcal{F}_i = \{S_{(1,i)}, \ldots, S_{(t_0,i)}\}$. For every vertex $v = (v_1, v_2, \ldots, v_m)$ of X_0 we set $\mathcal{G}(v) \stackrel{\text{def}}{=} \{S_{(v_1,1)}, \ldots, S_{(v_m,m)}\}$.

▷ Claim 12. $v \mapsto \mathcal{G}(v)$ is a bijection between the vertices of X_0 and the maximal colorful subfamilies of $\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_m$.

Let A be an axis-parallel k-dimensional affine subspace A, or *axis parallel k-flat* for short. We put $\mathcal{G}(A) \stackrel{\text{def}}{=} \bigcap_{v \in V(X_0) \cap A} \mathcal{G}(v)$.

▷ Claim 13. The map $A \mapsto \mathcal{G}(A)$ induces a bijection between the axis-parallel k-flats that intersect $V(X_0)$ and the colorful subfamilies of size $m-k$.

Last, we further associate to any chain $\alpha \in C_k(X_i)$ a colorful family $\mathcal{G}(\alpha)$ set to be $\mathcal{G}(A)$, where A is the smallest axis-parallel flat of $\mathbb{R}^m$ that contains the support of α (that is, its affine span). Note that if σ is a k-face of X_i, then $|\mathcal{G}(\alpha)| = m-k$.

6.2 A constrained chain map

The main technical step in the proof of Theorem 4 is the following construction.

▶ **Lemma 14.** *Under the conditions of Theorem 4, there exists a subgrid $\gamma : X_{\lceil d/2 \rceil} \hookrightarrow X_0$ and a nontrivial chain map $f_\# : C_\#\left((X_{\lceil d/2 \rceil})^{(\lceil d/2 \rceil)}\right) \to C_\#(\mathbb{R}^d)$ with the property that $\operatorname{supp} f_\#(\sigma) \subset \bigcap \mathcal{G}(\gamma_\#(\sigma))$ for any cell $\sigma \in (X_{\lceil d/2 \rceil})^{(\lceil d/2 \rceil)}$,*

Before we describe the construction of γ and $f_\#$, let us see how our weak colorful Helly theorem follows from Lemma 14.

Proof of Theorem 4. Let γ and $f_\#$ be as given by Lemma 14. Since $X_{\lceil d/2 \rceil} = G[d+3]^m$, we can apply Proposition 8 to find cells σ and τ in $(X_{\lceil d/2 \rceil})^{(\lceil d/2 \rceil)}$ such that:

1. $\dim \sigma + \dim \tau \leq d$, from Proposition 8,
2. σ and τ are not contained in any axis parallel hyperplane, from Proposition 8,
3. the supports of $f_\#(\sigma)$ and $f_\#(\tau)$ intersect, again from Proposition 8, and
4. $\operatorname{supp} f_\#(\sigma) \subset \bigcap \mathcal{G}(\gamma_\#(\sigma))$ and $\operatorname{supp} f_\#(\tau) \subset \bigcap \mathcal{G}(\gamma_\#(\tau))$, from Lemma 14.

From (3) and (4) it comes that there is a point contained in every member of $\mathcal{G}(\gamma_\#(\sigma)) \cup \mathcal{G}(\gamma_\#(\tau))$. From (2) and the definition of subgrids, it comes that the span of $\gamma_\#(\sigma)$ and the span of $\gamma_\#(\tau)$ are not contained in a common hyperplane. This in turn implies that $\mathcal{G}(\gamma_\#(\sigma))$ and $\mathcal{G}(\gamma_\#(\tau))$ are disjoint. Finally, we have

$$|\mathcal{G}(\gamma_\#(\sigma)) \cup \mathcal{G}(\gamma_\#(\tau))| = |\mathcal{G}(\gamma_\#(\sigma))| + |\mathcal{G}(\gamma_\#(\tau))| = (m - \dim\sigma) + (m - \dim\tau)$$

which is at least $2m-d$ by (1). ◀

6.3 Proof of Lemma 14

It remains to construct the announced subgrid γ and constrained chain map $f_\#$.

Proof of Lemma 14. We construct the subgrid $\gamma : X_{\lceil d/2 \rceil} \hookrightarrow X_0$ and the chain map $f_\#$ inductively using the subgrid lemma. For each $i = 0, 1, \ldots, \lceil d/2 \rceil$ we claim there exists a subgrid $\gamma^{(i)} : X_i \hookrightarrow X_0$ and a nontrivial chain map $f_\#^{(i)} : C_\# \left((X_i)^{(i)} \right) \to C_\#(\mathbb{R}^d)$ such that

$$\forall \sigma \in (X_i)^{(i)}, \quad \operatorname{supp} f_\#^{(i)}(\sigma) \subset \cap \mathcal{G} \left(\gamma_\#^{(i)}(\sigma) \right). \tag{8}$$

Setting $\gamma = \gamma^{(\lceil d/2 \rceil)}$ and $f_\# = f_\#^{\lceil d/2 \rceil}$ will then complete the proof.

For $i = 0$, we let $\gamma^{(0)}$ be the trivial inclusion $X_0 \hookrightarrow X_0$. For each vertex $v \in X_0$ we fix a point p_v in the intersection $\cap \mathcal{G}(v)$ of the maximal colorful family $\mathcal{G}(v)$, which is nonempty by hypothesis. We define the chain map $f_\#^{(0)}$ by setting $f_\#^{(0)}(v) = p_v$ for every vertex v of X_0.

Before proceeding to the inductive step, for each colorful subfamily $\mathcal{G}$ of $\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_m$ we fix a basis (arbitrarily) for $\widetilde{H}_i(\cap \mathcal{G})$, $0 \leq i \leq \lceil d/2 \rceil$. These bases remains fixed for remainder of the proof. The hypothesis $\phi_{\mathcal{G}}^{(\lceil d/2 \rceil)} \leq b$ allows to consider each homology group $\widetilde{H}_i(\cap \mathcal{G})$ as a subgroup of $(\mathbb{Z}_2)^b$.

Let $0 \leq i < \lceil d/2 \rceil$ and suppose we are given the subgrid $\gamma^{(i)} : X_i \hookrightarrow X_0$ and the chain map $f_\#^{(i)} : C_\# \left((X_i)^{(i)} \right) \to C_\#(\mathbb{R}^d)$ satisfying Condition (8). Let σ be an $(i+1)$-cell in X_i. The chain $\gamma_{\#i+1}^{(i)}(\sigma)$ is well-defined and has the same affine span as the chain $\gamma_{\#i}^{(i)}(\partial\sigma)$, so

$$\operatorname{supp} f_{\#i}^{(i)}(\partial\sigma) \subset \bigcap \mathcal{G} \left(\gamma_{\#i}^{(i)}(\partial\sigma) \right) = \bigcap \mathcal{G} \left(\gamma_{\#i+1}^{(i)}(\sigma) \right).$$

Define a homomorphism $h : C_{i+1}(X_i) \to (\mathbb{Z}_2)^b$ by setting

$$h(\sigma) \stackrel{\text{def}}{=} \left[f_{\#i}^{(i)}(\partial\sigma) \right] \in \widetilde{H}_i \left(\bigcap \mathcal{G} \left(\gamma_{\#i+1}^{(i)}(\sigma) \right) \right).$$

In other words, $h(\sigma)$ equals the homology class of the image $f_{\#i}^{(i)}(\partial\sigma)$ in the i-dimensional (reduced) homology group of $\cap \mathcal{G} \left(\gamma_{\#i+1}^{(i)}(\sigma) \right)$, which we can view as an element in $(\mathbb{Z}_2)^b$. By Claim 11, there exists a subgrid $\varphi : X_{i+1} \hookrightarrow X_i$ in the kernel of h. We set $\gamma^{(i+1)} \stackrel{\text{def}}{=} \gamma^{(i)} \circ \varphi$ and note that $\gamma^{(i+1)}$ is indeed a subgrid $X_{i+1} \hookrightarrow X_0$. Moreover, for every $(i+1)$-cell $\tau \in X_{i+1}$ we have $h(\varphi(\tau)) = 0$, that is $\left[f_{\#i}^{(i)}(\partial\varphi(\tau)) \right] = 0 \in \widetilde{H}_i \left(\bigcap \mathcal{G} \left(\gamma_{\#i+1}^{(i)}(\varphi(\tau)) \right) \right)$, which rewrites

$$\left[f_{\#i}^{(i)}(\varphi(\partial\tau)) \right] = 0 \in \widetilde{H}_i \left(\bigcap \mathcal{G} \left(\gamma_{\#i+1}^{(i+1)}(\tau) \right) \right). \tag{9}$$

For a cell $\sigma \in X_{i+1}$ of dimension at most i, we set $f_\#^{(i+1)}(\sigma) \stackrel{\text{def}}{=} f_\#^{(i)}(\varphi_\#(\sigma))$. For any $(i+1)$-cell $\tau \in X_{i+1}$, Equation (9) reveals that $f_\#^{(i+1)}(\partial\tau)$ is a boundary in $C_i \left(\bigcap \mathcal{G} \left(\gamma_{\#i+1}^{(i+1)}(\tau) \right) \right)$. We pick some arbitrary $\alpha \in C_{i+1} \left(\bigcap \mathcal{G} \left(\gamma_{\#i+1}^{(i+1)}(\tau) \right) \right)$ such that $\partial\alpha = f_\#^{(i+1)}(\partial\tau)$, and set $f_{\#i+1}^{(i+1)}(\tau) \stackrel{\text{def}}{=} \alpha$. Thus defined, $f_\#^{(i+1)}$ is indeed a chain map, from $C_\# \left((X_{i+1})^{(i+1)} \right)$ to $C_\# \left(\mathbb{R}^d \right)$, and it satisfies Condition (8). ◀

7 A stepping-up lemma for topological set systems

We can finally prove Theorem 1. Recall that we are given integers $d+1 \le k \le \ell$ and $b \ge 0$. Our task is to show that for any $\delta > 0$, there exists $\delta' > 0$ such that for any sufficiently large topological set system $\mathcal{F}$ in $\mathbb{R}^d$, if $\phi_{\mathcal{F}}^{(\lceil \frac{d}{2} \rceil)}(\ell) \le b$ and $\delta_{\mathcal{F}}(k) \ge \delta$, then $\delta_{\mathcal{F}}(\ell) \ge \delta'$.

Preparation. An m-uniform hypergraph is a pair $H = (V, E)$ where $V = V(H)$ is a finite set of vertices and $E = E(H) \subset \binom{V}{m}$ is the edge set. A hypergraph H *contains* a hypergraph H' if there is an injection $f : V(H') \to V(H)$ such that for every $e' \in E(H')$, $f(e) \in E(H)$. (In particular, we do *not* require that H' is an *induced* sub-hypergraph of H.) An m-uniform hypergraph is m-partite if the vertex set can be partitioned into disjoint sets (vertex classes) $V(H) = V_1 \cup \cdots \cup V_m$ such that every edge contains exactly one vertex from each V_i. Given integers $m \ge 2$ and $t \ge 1$, we let $K^m(t)$ denote the complete m-partite m-uniform hypergraph on vertex classes $V_1, \ldots, V_m$ where $|V_i| = t$. That is, the edge set of $K^m(t)$ consists of all m-tuples of $V_1 \cup \cdots \cup V_m$ that contain exactly one element from each V_i. We use the following "supersaturation" theorem of Erdős and Simonovits:

▶ **Theorem** ([13, Corollary 2]). *For any positive integers m and t and any $\varepsilon > 0$ there exists $\rho = \rho(\varepsilon, m, t) > 0$ such that any m-uniform hypergraph $H = (V, E)$ with $|E| \ge \varepsilon\binom{|V|}{m}$ contains at least $\rho|V|^{mt}$ copies of $K^m(t)$.*

Proof of Theorem 1. The general case follows from the special case where $\ell = k+1$ by stepping-up one dimension at a time.[2] Consider some topological set system $\mathcal{F}$ in $\mathbb{R}^d$. Let $t = t(b, d, k)$ be the constant from Theorem 4 where $b \stackrel{\text{def}}{=} \phi_{\mathcal{F}}^{(\lceil \frac{d}{2} \rceil)}(\ell)$ and the number m of colors is now k. For $\mathcal{F}' \subseteq \mathcal{F}$, let $H[\mathcal{F}']$ be the k-uniform hypergraph whose vertices are the members of $\mathcal{F}'$ and whose edges are the k-tuples of $\mathcal{F}'$ with nonempty intersection.

Now, our hypergraph $H[\mathcal{F}]$ contains at least $\delta\binom{|\mathcal{F}|}{k}$ edges. By the Erdős–Simonovits theorem it follows that for some constant $\rho > 0$ depending only on k, t, and δ, there are at least $\rho\binom{|\mathcal{F}|}{kt}$ distinct kt-element subfamilies $\mathcal{F}'$ of $\mathcal{F}$ such that $H[\mathcal{F}']$ contains a copy of $K^m(t)$. Our choice of t ensures that Theorem 4 applies to every such subfamily $\mathcal{F}'$, and therefore each $\mathcal{F}'$ contributes some $2k - d \ge k+1$ members with non-empty intersection. Each $(k+1)$-element subset of $\mathcal{F}$ with non-empty intersection is contained in $\binom{|\mathcal{F}|-(k+1)}{kt-(k+1)}$ distinct (kt)-tuples $\mathcal{F}'$. There are therefore at least

$$\frac{\rho\binom{|\mathcal{F}|}{kt}}{\binom{|\mathcal{F}|-(k+1)}{kt-(k+1)}} = \frac{\rho}{\binom{kt}{k+1}}\binom{|\mathcal{F}|}{k+1}$$

$(k+1)$-tuples of $\mathcal{F}$ with nonempty intersection. In other words, $\delta_{\mathcal{F}}(k+1)$ is at least $\delta' \stackrel{\text{def}}{=} \rho/\binom{kt}{k+1}$, where ρ depends only on k, t, and δ, that is on k, b, d and δ. ◀

[2] The careful reader may note that Theorem 4 allows to step up more than one dimension at a time, therefore weakening the assumption $\phi_{\mathcal{F}}^{(\lceil \frac{d}{2} \rceil)}(\ell) \le b$ to $\phi_{\mathcal{F}}^{(\lceil \frac{d}{2} \rceil)}\left(\ell'\right) \le b$ with $\ell' \stackrel{\text{def}}{=} \max\left(\lceil \frac{d+\ell}{2} \rceil, k\right)$.

References

1 N. Alon and G. Kalai. A simple proof of the upper bound theorem. *European Journal of Combinatorics*, 6(3):211–214, 1985.

2 N. Alon, G. Kalai, J. Matoušek, and R. Meshulam. Transversal numbers for hypergraphs arising in geometry. *Adv. in Appl. Math.*, 29(1):79–101, 2002.

3 N. Amenta. Helly theorems and generalized linear programming. *Discrete Comput. Geom.*, 12:241—-261, 1994.

4 I. Bárány. A generalization of Carathéodory's theorem. *Discrete Math.*, 40(2-3):141–152, 1982.

5 I. Bárány and J. Matoušek. A fractional Helly theorem for convex lattice sets. *Adv. Math.*, 174(2):227–235, 2003.

6 B. Bukh, J. Matoušek, and G. Nivasch. Lower bounds for weak epsilon-nets and stair-convexity. *Israel J. Math.*, 182:199–208, 2011.

7 S. Chakraborty, R. Pratap, S. Roy, and S. Saraf. Helly-type theorems in property testing. *International Journal of Computational Geometry & Applications*, 28(04):365–379, 2018.

8 L. Danzer, B. Grünbaum, and V. Klee. Helly's theorem and its relatives. In *Proc. Sympos. Pure Math., Vol. VII*, pages 101–180. Amer. Math. Soc., Providence, R.I., 1963.

9 J. De Loera, X. Goaoc, F. Meunier, and N. Mustafa. The discrete yet ubiquitous theorems of Carathéodory, Helly, Sperner, Tucker, and Tverberg. *Bulletin of the American Mathematical Society*, 56(3):415–511, 2019.

10 É. C. De Verdière, G. Ginot, and X. Goaoc. Helly numbers of acyclic families. *Adv. Math.*, 253:163–193, 2014.

11 J. Eckhoff. An upper-bound theorem for families of convex sets. *Geometriae Dedicata*, 19(2):217–227, 1985.

12 J. Eckhoff. Helly, Radon, and Carathéodory type theorems. In *Handbook of convex geometry, Vol. A, B*, pages 389–448. North-Holland, Amsterdam, 1993.

13 P. Erdős and M. Simonovits. Supersaturated graphs and hypergraphs. *Combinatorica*, 3(2):181–192, 1983. doi:10.1007/BF02579292.

14 X. Goaoc, P. Paták, Z. Patáková, M. Tancer, and U. Wagner. Bounding Helly numbers via Betti numbers. In *A journey through discrete mathematics*, pages 407–447. Springer, Cham, 2017.

15 R. L. Graham, B. L. Rothschild, and J. H. Spencer. *Ramsey theory*, volume 20. John Wiley & Sons, 1990.

16 S. Hell. Tverberg-type theorems and the fractional Helly property, 2006. PhD thesis.

17 A. Holmsen and D. Lee. Radon numbers and the fractional Helly theorem, 2019. arXiv:1903.01068.

18 A. F. Holmsen, M. Kim, and S. Lee. Nerves, minors, and piercing numbers. *Transactions of the American Mathematical Society*, 371(12):8755–8779, 2019.

19 T. Kaczynski, K. Mischaikow, and M. Mrozek. *Computational homology*, volume 157. Springer Science & Business Media, 2006.

20 G. Kalai. Intersection patterns of convex sets. *Israel Journal of Mathematics*, 48(2-3):161–174, 1984.

21 G. Kalai. Combinatorial expectations from commutative algebra. In I. Peeva and V. Welker, editors, *Combinatorial Commutative Algebra*, volume 1(3), pages 1729–1734. Oberwolfach Reports, 2004.

22 G. Kalai. Problems for Imre Bárány's birthday, 2017. URL: https://gilkalai.wordpress.com/2017/05/23/problems-for-imre-baranys-birthday/.

23 G. Kalai and R. Meshulam. A topological colorful Helly theorem. *Adv. Math.*, 191(2):305–311, 2005.

24 G. Kalai and R. Meshulam. Leray numbers of projections and a topological Helly-type theorem. *Journal of Topology*, 1(3):551–556, 2008.

25 G. Kalai and Z. Patáková. Intersection patterns of planar sets. *Discrete & Computational Geometry*, 64:304–323, 2020.

26 M. Katchalski and A. Liu. A problem of geometry in $\mathbb{R}^n$. *Proceedings of the American Mathematical Society*, 75(2):284–288, 1979.

27 J. Matoušek. A Helly-type theorem for unions of convex sets. *Discrete & Computational Geometry*, 18(1):1–12, 1997.

28 J. Matousek. *Lectures on discrete geometry*, volume 212. Springer Science & Business Media, 2013.

29 Z. Patáková. Bounding Radon Number via Betti Numbers. In Sergio Cabello and Danny Z. Chen, editors, *36th International Symposium on Computational Geometry (SoCG 2020)*, volume 164 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 61:1–61:13. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020.

30 U. Wagner. Minors in random and expanding hypergraphs. In *Proceedings of the 27th Annual Symposium on Computational Geometry (SoCG)*, pages 351–360, 2011.

31 Gerd Wegner. d-collapsing and nerves of families of convex sets. *Archiv der Mathematik*, 26(1):317–321, 1975.

Throwing a Sofa Through the Window

Dan Halperin
Tel Aviv University, School of Computer Science, Israel

Micha Sharir
Tel Aviv University, School of Computer Science, Israel

Itay Yehuda
Tel Aviv University, School of Computer Science, Israel

Abstract

We study several variants of the problem of moving a convex polytope K, with n edges, in three dimensions through a flat rectangular (and sometimes more general) window. Specifically:

(i) We study variants where the motion is restricted to translations only, discuss situations where such a motion can be reduced to sliding (translation in a fixed direction), and present efficient algorithms for those variants, which run in time close to $O(n^{8/3})$.

(ii) We consider the case of a *gate* (an unbounded window with two parallel infinite edges), and show that K can pass through such a window, by any collision-free rigid motion, iff it can slide through it, an observation that leads to an efficient algorithm for this variant too.

(iii) We consider arbitrary compact convex windows, and show that if K can pass through such a window W (by any motion) then K can slide through a slab of width equal to the diameter of W.

(iv) We show that if a purely translational motion for K through a rectangular window W exists, then K can also slide through W keeping the same orientation as in the translational motion. For a given fixed orientation of K we can determine in linear time whether K can translate (and hence slide) through W keeping the given orientation, and if so plan the motion, also in linear time.

(v) We give an example of a polytope that cannot pass through a certain window by translations only, but can do so when rotations are allowed.

(vi) We study the case of a circular window W, and show that, for the regular tetrahedron K of edge length 1, there are two thresholds $1 > \delta_1 \approx 0.901388 > \delta_2 \approx 0.895611$, such that (a) K can slide through W if the diameter d of W is ≥ 1, (b) K cannot slide through W but can pass through it by a purely translational motion when $\delta_1 \leq d < 1$, (c) K cannot pass through W by a purely translational motion but can do it when rotations are allowed when $\delta_2 \leq d < \delta_1$, and (d) K cannot pass through W at all when $d < \delta_2$.

(vii) Finally, we explore the general setup, where we want to plan a general motion (with all six degrees of freedom) for K through a rectangular window W, and present an efficient algorithm for this problem, with running time close to $O(n^4)$.

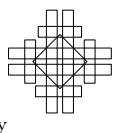
2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Motion planning, Convex polytopes in 3D

Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.41

Related Version *Full Version*: `https://arxiv.org/abs/2102.04262`

Funding Work on this paper by DH and IY has been supported in part by the Israel Science Foundation (grant no. 1736/19), by NSF/US-Israel-BSF (grant no. 2019754), by the Israel Ministry of Science and Technology (grant no. 103129), by the Blavatnik Computer Science Research Fund, and by a grant from Yandex. Work by MS and IY has been supported in part by Grant 260/18 from the Israel Science Foundation. Work by MS has also been supported by Grant G-1367-407.6/2016 from the German-Israeli Foundation for Scientific Research and Development, and by the Blavatnik Research Fund in Computer Science at Tel Aviv University.



Acknowledgements We deeply thank Pankaj Agarwal and Boris Aronov for useful interactions concerning this work. In particular, Boris has suggested an alternative proof of the key topological property in the analysis in Section 4, and Pankaj has been instrumental in discussions concerning the range searching problem in Section 2. We are also grateful to Lior Hadassi for interaction involving the topological property presented in Section 4, and to Eytan Tirosh for help with the alternative proof of Lemma 1.

1 Introduction

Let K be a convex polytope (a "sofa") in $\mathbb{R}^3$ with n edges, and let W be a rectangular window, placed in the xy-plane in the axis-parallel position $[0, a] \times [0, b]$, where a and b are the respective width and height of W. We assume that the complement of W in the xy-plane is a solid wall that K must avoid. The problem is to determine whether K can be moved, in a collision-free manner, from any position that is fully contained in the upper halfspace $z > 0$, through W, to any position that is fully contained in the lower halfspace $z < 0$, and, if so, to plan such a motion (see Figure 1).

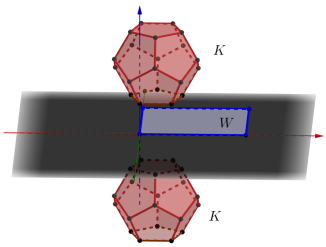


Figure 1 Moving a convex polytope K through a window W.

A continuous motion of a rigid body in three dimensions has six degrees of freedom, three of translation and three of rotation, and in the general form of the problem, studied in Section 8, we allow all six degrees. On the way, we will study simpler versions where only restricted types of motion are allowed, such as purely translational motion (that has only three degrees of freedom), a translational motion in a fixed direction, which we refer to as *sliding* (one degree of freedom), or a translational motion combined with rotations around the vertical axis only (four degrees of freedom), etc. Some of our main results show that, in certain favorable situations, the existence of a general collision-free motion of K through W implies the existence of a restricted motion of one of these types. This allows us to solve the problem in a significantly more efficient manner.

In terms of the *free configuration space* $\mathcal{F}$ of K, all the placements of K that are fully contained in the upper (resp., lower) halfspace are free, and form a connected subset $\mathcal{F}^+$ (resp., $\mathcal{F}^-$) of $\mathcal{F}$. Our problem, in general, is to determine whether both $\mathcal{F}^+$ and $\mathcal{F}^-$ are contained in the same connected component of $\mathcal{F}$. This interpretation applies to the general setup, with six degrees of freedom, as well as to any other subclass of motions, with fewer degrees of freedom.

Motion planning is an intensively studied problem in computational geometry and robotics. A systematic and general way to describe the free space $\mathcal{F}$ is by using *constraint surfaces*, namely surfaces describing all the configurations where some feature on the boundary of the moving object (K in our case) touches a feature on the boundary of the free workspace (W in our case); see, e.g., [12, 18, 19]. These surfaces partition the configuration space into cells, each of which is either fully contained in $\mathcal{F}$ or fully contained in the forbidden portion of the

configuration space. This representation is based on the arrangement $\mathcal{A}$ of the constraint surfaces, whose number in our case is $O(n)$, one surface for each feature (edge or vertex) of W and each feature (edge, vertex or face) of K. Each cell of $\mathcal{A}$ is fully contained either in $\mathcal{F}$ or in its complement. Hence the complexity of $\mathcal{F}$ is $O(n^d)$, where d is the number of degrees of freedom (namely, the dimension of the configuration space) [12]. To exploit this representation, we construct and transform it into a so-called discrete *connectivity graph*, which can be searched for the existence of a motion of the desired kind.

One common way of doing this is to further decompose the arrangement into subcells of constant complexity, using *vertical decomposition* [4]. Such constructions are easily implementable for motion planning with two degrees of freedom [8], but become significantly more involved for problems with three or more degrees of freedom. This has lead to the development of alternative methods, such as sampling-based techniques (see [5, Chapter 7] and [10]), the best known of which are PRM [16] and RRT [17], which have dozens of variants. While extremely successful in solving practical problems, they trade-off the completeness of the arrangement approach with efficiency, and may fail when the setting contains tight passages [21, 22], a situation that can arise in the problems that we study in this paper.

Toussaint [25] studied movable separability of sets, where he collected a variety of tight-setting motion planning problems, similar in nature to the problems studied here. These problems are interesting theoretically (see, e.g., [24] for such a problem and its intriguing solution), but also from an applied perspective, since motion in tight settings often arises in manufacturing processes such as *assembly planning* [11] or *casting and molding* [3].

It was in Toussaint's review [25] that we encountered the problem of "throwing" a polytope through a window. Although Toussaint's paper was published 35 years ago, we are not aware of any previous progress on this specific problem. We remark that the word *sofa* in the title of the paper is borrowed from the classical two-dimensional *moving sofa* problem (see, e.g., [6, 9]), which is to find the shape of largest area that can be moved through a corner in an L-shaped corridor whose legs have width 1.

Our results. We first consider, in Section 2, sliding motions (translations in a fixed direction) of K. We characterize situations in which such a sliding motion exists, and present efficient algorithms, with runtime close to $O(n^{8/3})$, for finding such a motion when one exists.

We next consider, in Section 3, the case where W is an unbounded slab, enclosed between two parallel unbounded lines (we call it a *gate*). We show that if K can pass through such a gate W, by any collision-free rigid motion, it can also slide throgh W, making the general motion planning problem through a gate particularly easy to solve.

In Section 4, we consider arbitrary compact convex windows, and show that if K can move through such a winodw W, by an arbitrary collision-free motion, then K can slide through a gate of width equal to the diameter of W, and this holds in any sliding direction. This requires nontrivial topological arguments, presented in Section 4 and in the full version of the paper [14].

We then consider, in Section 5, purely translational motions of K through a rectangular window W, and prove that the existence of such a purely-translational collision-free motion implies the existence of a collision-free sliding motion keeping the same orientation as in the translational motion. For a given fixed orientation of K we can determine in linear time whether K can translate (and hence slide) through W keeping the given orientation, and if so plan the motion, also in linear time. We also give a near-linear time algorithm for planning the motion of K through an arbitrary (flat) polygonal window of fixed (constant) complexity.

In Section 6, we show that rotations are sometimes needed, by giving an example of a convex polytope K (actually a tetrahedron) that can move through a square window W by a collision-free motion that includes rotation (only around an axis orthogonal to W), but there is no purely translational motion of K through W.

In Section 7, we consider the problem of passing through a circular window W, and show that, for the regular tetrahedron K of edge length 1, there are two thresholds $1 > \delta_1 \approx 0.901388 > \delta_2 \approx 0.895611$, such that (i) K can slide through the window W if the diameter d of W is ≥ 1, (ii) K cannot slide through W but can pass through it by a purely translational motion when $\delta_1 \leq d < 1$, (iii) K cannot pass through W by a purely translational motion but can do it with rotations when $\delta_2 \leq d < \delta_1$, and (iv) K cannot pass through W at all when $d < \delta_2$.

We finally consider, in Section 8, the general problem, with all six degrees of freedom. We present an efficient algorithm, which runs in time close to $O(n^4)$, for constructing the free configuration space, from which one can construct, within a comparable time bound, a valid motion through W if one exists.

2 Translation in a fixed direction

In this section we address the case in which the movement is purely translational in a single fixed direction. Such a motion, to which we refer as a *sliding motion*, has only one degree of freedom. In the most restricted version (which is very easy to solve), we are given a fixed orientation of K at a fixed initial placement, and also the direction of motion. In this section we study a more general setting, in which we seek values for these parameters – orientation, initial placement, and direction of motion, for which such a sliding motion of K through W is possible (or determine that no such motion is possible).

In Section 2.1 we observe that if a sliding motion for K exists, then K can also slide in a direction orthogonal to the plane of the window. Using this and other structural properties of the problem, we transform the problem at hand into a certain range searching problem. We present an efficient novel solution to the latter problem, which yields an algorithm for solving our original problem, whose running time is close to $O(n^{8/3})$.

2.1 The existence of an orthogonal sliding motion

For the most general version of the sliding motion, in which none of the parameters (orientation, initial placement, and direction of motion) is prespecified, we have:

▶ **Lemma 1.** *If K can slide through W from some starting placement in some direction, then K can slide through W, possibly from some other starting placement (and at another orientation), by translating it in the negative z-direction.*

Proof. Let K_0 be the starting placement of K and let $\vec{v}$ be the direction of motion through W, for which the resulting sliding motion is collision-free. Form the infinite prism $\Pi_0 := \bigcup_{\lambda \in \mathbb{R}} (K_0 + \lambda \vec{v})$ that K_0 spans in direction $\vec{v}$. The premise of the lemma implies that the intersection of Π_0 with the xy-plane is contained in W.

Let W_0 be the orthogonal projection of W onto some plane orthogonal to $\vec{v}$. Note that W_0 is a parallelogram, and that, by construction, K_0 can pass through W_0 when translated in direction $\vec{v}$. By an old result, reviewed and proved by Debrunner and Mani-Levitska [7], it follows that, when mapped rigidly into the xy-plane, W_0 (the "shadow" of W in direction $\vec{v}$) can be placed fully within W (see Figure 2).[1]

[1] Curiously, as shown in [7], this property, of containing your shadows, may fail in higher dimensions.

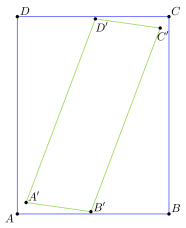


Figure 2 The projection of W (green) can be located in a congruent copy of W (blue).

Now rotate and translate $\mathbb{R}^3$ so that $\vec{v}$ becomes the (negative) z-direction, and the image of W_0 is fully contained in (the former, untransformed copy of) W. Then the image of K under this transformation can be moved vertically down through W, in a collision-free manner, as asserted. ◀

Debrunner and Mani-Levitska's proof is involved, and applies to an arbitrary planar convex shape (showing that it contains its projection in any direction). In the full version of the paper [14] we give a simple alternative proof for the case of a rectangle.

2.2 Finding a sliding motion

The following discussion is with respect to a fixed initial placement K_0 of K. For a given direction $\vec{v}$, the *projected silhouette* of K_0 in direction $\vec{v}$ is the boundary of the convex polygon obtained by the projection of K_0 in direction $\vec{v}$, within the image plane $h_{\vec{v}}$ (which is orthogonal to $\vec{v}$). The silhouette itself is the cyclic sequence of vertices and edges of K, whose projections form the projected silhouette.[2] The silhouette and its projection do not change combinatorially, that is, when represented as a cyclic sequence of vertices and edges of K (or of their projections), as long as $\vec{v}$ is not parallel to any face of K_0. We thus form the set of the $O(n)$ great circles on $\mathbb{S}^2$ that are parallel to the faces of K_0, and construct their arrangement $\mathcal{A}_0$ on $\mathbb{S}^2$, which is also known as the *aspect graph* of K_0 [20]. In each face φ of $\mathcal{A}_0$ the combinatorial structure of the silhouette is fixed, but the projected silhouette varies continuously as $\vec{v}$ moves in φ.

A *view* of K_0 is a pair $(\vec{v}, \theta)$, where $\vec{v}$ is a direction, and θ is the angle of rotation of the projected silhouette within $h_{\vec{v}}$ (translations within that plane are ignored). The space of views is thus three-dimensional. A fixed view $(\vec{v}, \theta)$ fixes the uppermost, leftmost, bottommost and rightmost vertices w_t, w_l, w_b and w_r of the projected silhouette. The view is valid if

$$x_{v_r} - x_{v_l} \leq a \qquad \text{and} \qquad y_{v_t} - y_{v_b} \leq b \tag{1}$$

(in the coordinate frame of $h_{\vec{v}}$ when rotated by θ).

When $\vec{v}$ is fixed and θ varies, we get $O(n)$ quadruples (w_t, w_l, w_b, w_r) of the projected silhouette. The view is valid if the inequalities in (1) (which depend on $(\vec{v}, \theta)$) have a solution for one such quadruple, which lies in the appropriate portion of the view space (in which w_t, w_l, w_b, w_r are indeed the four extreme vertices). The existence of a valid view is equivalent to the existence of a sliding motion of K_0 through W, after suitably shifting K_0 and rotating it around $\vec{v}$ by θ. We give more details in the full version [14], where we show that the total number of quadruples of vertices is $O(n^3)$, from which we obtain an algorithm for finding a valid view, with near-cubic running time.

[2] The silhouette is indeed such a cycle of vertices and edges of ∂K for generic directions $\vec{v}$. When $\vec{v}$ is parallel to a face f of K_0, the entire f is part of the silhouette.

2.3 An improved algorithm

We next present an improved, albeit more involved algorithm that solves the problem of finding a sliding motion of K, if one exists, in time $O(n^{8/3}\text{polylog}(n))$. The problem of finding a direction $\vec{v}$ in which we can slide K through W is equivalent to the problem of finding a placement of W in some plane h, such that the projected silhouette of K on h is contained in W, which in turn is equivalent to verifying that all the projected vertices of K fit into that placement of W.

An equivalent way of checking for the latter characterization is to look for two unit vectors x and y in $\mathbb{R}^3$ (which will be the directions of the axes of W in the desired placement; note that h is spanned by x and y) that satisfy:

(i) x and y are perpendicular to each other.

(ii) For every segment e connecting two vertices of K we have $\langle x, e\rangle \leq a$.

(iii) For every segment e connecting two vertices of K we have $\langle y, e\rangle \leq b$.

(Note that since we go over all ordered pairs of vertices of K in (ii), (iii), we actually require that $|\langle x, e\rangle| \leq a$ and $|\langle y, e\rangle| \leq b$ for each such segment e.) Every inequality in (ii) defines a halfspace that has to contain x. We intersect those $O(n^2)$ halfspaces, to obtain a convex polytope Q of complexity $O(n^2)$, and intersect Q with the unit sphere $\mathbb{S}^2$ to obtain the admissible region A of the vectors x that satisfy (ii), in $O(n^2 \log n)$ time. The region A is bounded by circular arcs (not necessarily arcs of great circles), which meet at the *vertices* of A. We apply the same procedure for y using the suitable collection of halfspaces in (iii), and obtain the admissible region B for the vectors y that satisfy (iii), also in $O(n^2 \log n)$ time.

To satisfy (i) too, we need to check whether there exist an orthogonal pair of vectors $x \in A$, $y \in B$. We use the following lemma, whose proof is given in the full version [14].

▶ **Lemma 2.** *Let S_A denote the set of all vertices of A, and let T_A denote the set of the points that are closest locally to the north pole of $\mathbb{S}^2$ along each circular arc of ∂A. (If the direction of the north pole of $\mathbb{S}^2$ is generic, T_A is finite and $|S_A \cup T_A| = O(n^2)$.) Define similarly the sets S_B, T_B. If there exist an orthogonal pair $(x, y) \in A \times B$ then there exist such an orthogonal pair so that either $x \in S_A \cup T_A$ or $y \in S_B \cup T_B$.*

We iterate over the points of $S_A \cup T_A$. For each such point v let C_v be the great circle orthogonal to v, and let $\mathcal{C}$ denote the collection of these $O(n^2)$ great circles. We face the problem of determining whether any great circle in $\mathcal{C}$ crosses B, which is essentially the problem of determining whether any great circle in $\mathcal{C}$ crosses an arc of ∂B. By centrally projecting the setup onto two parallel planes, the problem is further reduced to the problem where we have a set of $M = O(n^2)$ lines and a set of $N = O(n^2)$ arcs of conic sections in the plane, and the goal is to determine whether there is a line-arc intersection. The important role of Lemma 2 is that it enables us to reduce the quest for an orthogonal pair (x, y) into a finite number ($O(n^2)$) of range-searching queries, by asserting that if an orthogonal pair exists, then there exists an orthogonal pair where at least one of x, y is from a prescribed discrete set of points.

In the full version [14] we give full details of our solution. Here is a brief sketch. With no loss of generality, assume that each input arc is x-monotone and convex. Then a rightward-oriented line ℓ intersects an arc γ if and only if one of the following conditions holds:

(i) The two endpoints of γ lie on different sides of ℓ. See Figure 3(i).

(ii) The two endpoints of γ lie to the left of ℓ, γ has a tangent that is parallel to ℓ (that is, the slope a of ℓ lies between the slopes of the tangents to γ at its endpoints), and ℓ lies to the left of the tangent to γ with slope a. See Figure 3(ii).

Using carefully designed multi-level range-searching data structures, we can test whether some pair of an input line and an input arc satisfy (i) or (ii), in time $O(M^{2/3}N^{2/3}\text{polylog}(M+N)) = O(n^{8/3}\text{polylog}(n))$.[3] Hence, within the same time bound, we can find a valid view of K, and thus a collision-free sliding motion of K through W, if one exists. That is, we have:

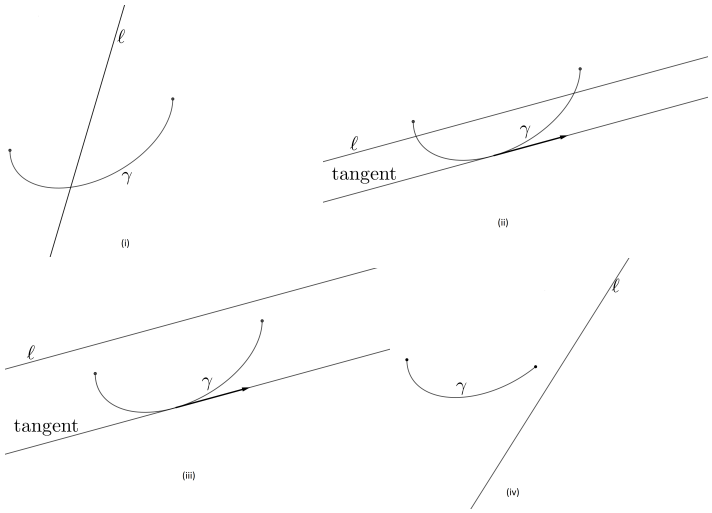


Figure 3 A line ℓ intersecting a convex x-monotone elliptic arc γ: (i) The two endpoints of γ lie on different sides of ℓ. (ii) The two endpoints lie to the left of ℓ and ℓ lies to the left of the parallel tangent to the arc. (iii) The two endpoints lie to the right of ℓ (and then there is no intersection). (iv) The two endpoints lie to the left of ℓ but γ has no tangent parallel to ℓ (and then there is no intersection).

▶ **Theorem 3.** *Given K and W as above, we can determine whether K can slide through W in a collision-free manner, and, if so, find such a sliding motion, in time $O(n^{8/3}\text{polylog}(n))$.*

3 Unbounded windows

In this section we consider the variant in which W is an infinite slab in the xy-plane, bounded by, say, two vertical lines $x = 0$ and $x = a$. We refer to such a window as a *gate*. We show:

▶ **Theorem 4.** *Let K be a convex polytope that can be moved by some collision-free rigid motion through a gate W. Then there exists a sliding collision-free motion of K through W.*

We can therefore apply the machinery of Theorem 3, and conclude that we can determine whether K can be moved through W by a collision-free motion, in time $O(n^{8/3}\text{polylog}(n))$.

The full details of the proof of the theorem are given in the full version [14]. Here is brief sketch. By projecting the moving polytope K and W onto the xz-plane, W projects to the interval $g := [0, a] \times \{0\}$, and K projects to a time-varying convex polygon that starts from a placement that lies in the upper halfplane $z > 0$ and reaches a placement that lies in the lower halfplane $z < 0$. For technical reasons, we approximate K by a smooth convex body, and reduce the problem to the case where K is smooth and convex.

[3] We are not aware of any published result that solves this problem efficiently. A different solution, with a similar performance bound, was improvised and sketched to us by Pankaj Agarwal, and we thank him deeply for the useful interaction concerning this problem.

At any time t during the motion, the projected planar region $\pi(K(t))$ (where $K(t)$ is the placement of K at time t) meets g at some interval $I(t)$ (we ignore the prefix and suffix of the motion where $K(t)$ does not yet meet, or no longer meets W). We consider the two tangents to $\pi(K(t))$ at the endpoints $\kappa^-(t)$, $\kappa^+(t)$ of $I(t)$, and note that, at the beginning of the motion, the wedge that these tangents form and that contains $K(t)$ points upwards, and at the end of the motion it points downwards; see Figure 4.

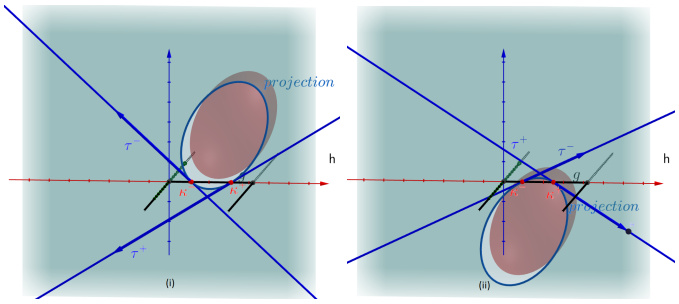


Figure 4 Moving the projection of K through g. Left: At the beginning of the crossing of g, the tangents $\tau^-(t)$ and $\tau^+(t)$ "open up" (with respect to their sides that contain $K(t)$). Right: At the end of the crossing, they "open down".

Since the tangents vary continuously (because $K(t)$ is always smooth), there must be a time t_0 at which the two tangents are parallel to each other, and thus span a slab σ (in the xz-plane) whose width is clearly $\leq a$. See Figure 5. The Cartesian product of σ and the y-axis yields a slab σ^* in $\mathbb{R}^3$, whose cross-section with the xy-plane is contained in W. This in turn implies that K can slide through W, and completes the proof. □

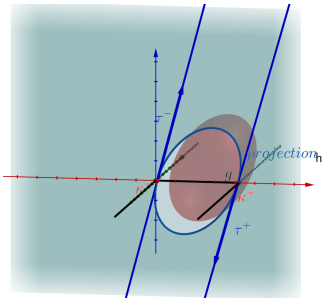


Figure 5 The critical instance t_0 where the tangents at $\kappa^-(t_0)$ and at $\kappa^+(t_0)$ become (anti-) parallel.

4 From passing through an arbitrary convex window to sliding through a gate

In this section we prove a similar yet different property of a convex polytope passing through an arbitrary compact planar convex window, not necessarily rectangular.

▶ **Theorem 5.** *Let W be an arbitrary compact convex region in the xy-plane. Let K be a convex polytope that can be moved by some collision-free motion (possibly full rigid motion, with six degrees of freedom) through W, and let d be the diameter of W (the maximum distance between any pair of points in W). Let h be an arbitrary plane, and let K_h be the*

orthogonal projection of K on h. Then K_h can be rigidly placed between two parallel lines at distance d. That is, for any fixed direction $\vec{v}$, K can slide, from its (arbitrary) initial placement, in direction $\vec{v}$ through a gate of width d, in a plane perpendicular to $\vec{v}$.

We provide two different topology-based proofs of Theorem 5, both presented in full detail in the full version [14]. We sketch here one of these proofs. But first here is an interesting corollary of the theorem, also proved in [14].

▶ **Corollary 6.** *If K can be moved through a rectangular window W of dimensions $a \times b$ by some collision-free motion, then K can slide through a rectangle of dimensions* $\min(a,b) \times \sqrt{a^2+b^2}$.

Sketch of a proof of Theorem 5. Similar to the previous section, we first prove the theorem for smooth strongly convex compact bodies, and then extend the result to polytopes (see [14]). Consider the motion of K, now assumed to be a smooth, strongly convex, and compact body, during the time interval $[0,1]$. Assume that at $t=0$ (resp., at time $t=1$), K lies fully above (resp., below) the xy-plane.

Fix some direction $\vec{v}$, and let $C = C(\vec{v})$ denote the silhouette of K when viewed in direction $\vec{v}$. Let h be some plane orthogonal to $\vec{v}$, and let π_h denote the orthogonal projection onto h. Parameterize a point $u \in C$ by the orientation θ of the tangent at $\pi_h(u)$ to $K_h := \pi_h(K)$, which is well defined since K is smooth and strongly convex, and let $\gamma = \gamma_h$ be the inverse of π_h, that is, $\gamma(\theta)$ is the unique point $u \in C$ such that $\pi_h(u) = \theta$. Since K is assumed to be strongly convex, K_h is also strongly convex, and γ is a well-defined and continuous function on $\mathbb{S}^1$. We extend γ to a bivariate function $\gamma^* : \mathbb{S}^1 \times [0,1] \mapsto \mathbb{R}^3$, so that $\gamma^*(\theta,t)$ is the position (in the ambient 3-space) of $\gamma(\theta)$ at time t during the motion of K.

Let $\delta : \mathbb{S}^1 \times [0,1] \mapsto \mathbb{R}$ be the function $\delta(\theta,t) = z(\gamma^*(\theta,t))$, namely, the z-coordinate of the corresponding point $\gamma(\theta)$ of C at time t. Note that at time $t=0$ (resp., $t=1$), δ is positive (resp., negative) at each θ, since K lies fully above (resp., below) the xy-plane at that time. Put $M := \max_{\theta \in \mathbb{S}^1} \delta(\theta,0)$ and $m := \min_{\theta \in \mathbb{S}^1} \delta(\theta,1)$, so $M > 0$ and $m < 0$.

The functions $\delta_0(\theta) = \delta(\theta,0)$ and $\delta_1(\theta) = \delta(\theta,1)$ are defined and continuous on $\mathbb{S}^1$, and we extend each of them to the closed unit disk $\mathcal{B}^1$ bounded by $\mathbb{S}^1$, in polar coordinates, which, for technical reasons, we write in reverse order as (θ, r), by

$$\delta_0^*(\theta,r) = r\delta_0(\theta) + (1-r)M$$
$$\delta_1^*(\theta,r) = r\delta_1(\theta) + (1-r)m.$$

It is easily checked that these extensions are well defined and continuous over $\mathcal{B}^1$. Moreover, $\delta_0^*(\theta,r) > 0$ and $\delta_1^*(\theta,r) < 0$ for every θ.

We now take our function δ, which is so far defined on the side surface S of the cylinder $\mathbb{S}^1 \times [0,1]$, and extend it to the entire boundary $S^* := S \cup B_0 \cup B_1$ of the cylinder, so that δ coincides with δ_0^* on the base B_0 of the cylinder at $t=0$, and with δ_1^* on the base B_1 at $t=1$. Clearly, the extended δ is well defined and continuous over S^*.

To simplify the forthcoming analysis, we identify S^* with the unit sphere $\mathbb{S}^2$, which we parameterize by (θ, z), where $\theta \in \mathbb{S}^1$ is the horizontal orientation of the point on $\mathbb{S}^2$ and z is its z-coordinate (so θ is not well defined at the north and south poles of $\mathbb{S}^2$). We use the simple homeomorphism f that maps a point $(\theta,t) \in S$ to $(\theta, t-1/2) \in \mathbb{S}^2$, maps a point $(\theta,r) \in B_0$ to $(\theta, -1+r/2) \in \mathbb{S}^2$, and maps a point $(\theta,r) \in B_1$ to $(\theta, 1-r/2) \in \mathbb{S}^2$. See Figure 6 for an illustration.

Define a function G from $\mathbb{S}^2$ to $\mathbb{R}^2$ by $G(\theta,z) = (\delta(\theta,z),\ \delta(\theta+\pi,z))$, for $(\delta,z) \in \mathbb{S}^2$. Our goal is to show that $G(\mathbb{S}^2)$ contains the origin. Note that, by construction, $G(f(B_0))$ is fully contained in the positive quadrant $Q_1 := \{(x,y) \mid x,y > 0\}$, and $G(f(B_1))$ is fully contained

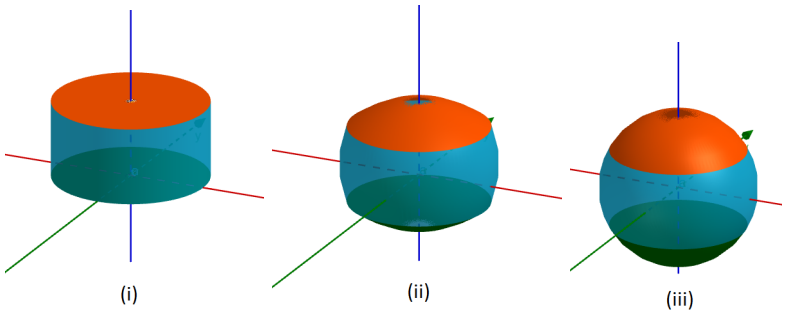


Figure 6 Identifying S^* with the unit sphere $\mathbb{S}^2$. B_0 is shown in green, B_1 in orange, and S in light blue. In (i) S^* is depicted, in (ii) an intermediate snapshot of the deformation is shown, for visual convenience, and in (iii) the final unit ball is shown, divided into the three parts that correspond to B_0, S and B_1.

in the negative quadrant $Q_3 := \{(x, y) \mid x, y < 0\}$. Thus, if $G(\mathbb{S}^2)$ contains the origin then so does $G(f(S))$. Once this property is established, it provides us with a pair (θ, z) such that $\delta(\theta, z) = \delta(\theta + \pi, z) = 0$, which means that there are two antipodal points $u, v \in C$ that pass through W simultaneously. Therefore their distance must be at most the diameter of W, and hence also the distance between the parallel tangent planes through them, which is a slab parallel to $\vec{v}$ of width at most d that contains K, as asserted.

Assume to the contrary that $G(\mathbb{S}^2)$ does not contain the origin. Then we can normalize G to the function

$$H(\theta, z) := \frac{G(\theta, z)}{\|G(\theta, z)\|}, \qquad \text{for } (\theta, z) \in \mathbb{S}^2,$$

which maps $\mathbb{S}^2$ continuously to the unit circle $\mathbb{S}^1$. The function G, and thus also the function H, are symmetric with respect to the line $y = x$ in $\mathbb{R}^2$, meaning that

$$G(\theta + \pi, z) = \Sigma(G(\theta, z)), \qquad \text{for } (\theta, z) \in \mathbb{S}^2, \quad \text{and thus also}$$
$$H(\theta + \pi, z) = \Sigma(H(\theta, z)), \qquad \text{for } (\theta, z) \in \mathbb{S}^2,$$

where Σ is the reflection about $y = x$, that is, $\Sigma(x, y) = (y, x)$.

We now use the property that the real line is a *covering space* of $\mathbb{S}^1$, in the specific (and easily verified) sense that the continuous map $p : \mathbb{R} \mapsto \mathbb{S}^1$, given by $p(x) = e^{2\pi i x}$, for $x \in \mathbb{R}$, is surjective, and, for each $w \in \mathbb{S}^1$, there exists an open neighborhood U of w such that $p^{-1}(U)$ is the disjoint union of open sets in $\mathbb{R}$, each of which is mapped homeomorphically to U by p. The map p is called the *covering map*.

A well known property of covering spaces is the *lifting property* (reviewed, e.g., in [1]; see also [15]), a special case of which asserts, in the specific context used here, that, if φ is any continuous map from $\mathbb{S}^2$ to $\mathbb{S}^1$ then φ can be *lifted* to a map $\psi : \mathbb{S}^2 \mapsto \mathbb{R}$, so that $p \circ \psi = \varphi$.

Applying the lifting property to the function H, we get a continuous mapping $T : \mathbb{S}^2 \mapsto \mathbb{R}$, such that $p \circ T = H$, so we have the property that

$$p(T(\theta + \pi, z)) = \Sigma(p(T(\theta, z))), \qquad \text{for } (\theta, z) \in \mathbb{S}^2.$$

As is easily checked, we have $\Sigma\left(e^{iy}\right) = e^{i(\pi/2 - y)}$, and therefore, for a point $x \in \mathbb{R}$, we have

$$\Sigma(p(x)) = \Sigma\left(e^{2\pi i x}\right) = e^{\pi i/2 - 2\pi i x} = p(1/4 - x), \qquad \text{so}$$
$$p(T(\theta + \pi, z)) = p(1/4 - T(\theta, z)), \qquad \text{for } (\theta, z) \in \mathbb{S}^2.$$

This in turn implies, by the definition of p, that

$$T(\theta+\pi,z) = 1/4 + k_{\theta,z} - T(\theta,z),$$

for some integer $k_{\theta,z}$. However, since T is continuous, there must be a single integer k such that $k_{\theta,z} \equiv k$ for all θ and z. That is, we have

$$T(\theta+\pi,z) + T(\theta,z) = 1/4 + k, \qquad \text{for all } (\theta,z) \in \mathbb{S}^2. \tag{2}$$

By an easy application of the mean-value theorem (which is also a special case of the Borsuk-Ulam theorem in dimension 1), there exist θ_0 and θ_1 such that, recalling that the value $z = -1/2$ (resp., $z = 1/2$) corresponds to points on the lower (resp., upper) circle bounding S,

$$\begin{aligned} T(\theta_0+\pi,-1/2) &= T(\theta_0,-1/2) \\ T(\theta_1+\pi,1/2) &= T(\theta_1,1/2). \end{aligned}$$

Substituting in (2), we get

$$T(\theta_0,-1/2) = T(\theta_1,1/2) = 1/8 + k/2.$$

However, by construction, $H(\theta_0,-1/2)$ lies in the first quadrant Q_1, and $H(\theta_1,1/2)$ lies in the third quadrant Q_3. Hence we have $T(\theta_0,-1/2) \in (0,1/4)+\mathbb{Z}$ and $T(\theta_1,1/2) \in (1/2,3/4)+\mathbb{Z}$, but $1/8+k/2$ can belong to only one of these sets (depending on whether k is even or odd). This contradiction shows that $G(\mathbb{S}^2)$, and thus also $G(f(S))$, contains the origin, as asserted.

So far the proof was for a smooth strongly convex compact bodies. The extension to the case of a convex polytope K is done exactly as in the analysis in the preceding section (which is spelled out in the full version [14]). ◀

5 Purely translational motions

In this section we show that purely translational motions of K through a rectangular window W are not more powerful than sliding. Specifically, we have the following theorem, which is, in a sense, a strengthening of Lemma 1. The proofs of the theorems in this section are given in [14].

▶ **Theorem 7.** *If K can be moved through a rectangular window W by a purely translational collision-free motion in some fixed orientation Θ, then K can be moved through W, possibly from some other starting position, by sliding while keeping the same orientation Θ.*

We also address a more restricted case where we are given a prescribed orientation Θ and we wish to find a purely translational motion for K with this orientation. We denote the polytope K at orientation Θ (ignoring translations) by K^Θ. We obtain the following results.

▶ **Theorem 8.** *Given an orientation Θ, we can determine whether a translational motion for K^Θ through the rectangular W exists, and if so find a sliding motion for K^Θ through W, in $O(n)$ time.*

▶ **Theorem 9.** *Let W be an arbitrary (not necessarily convex) polygonal window with k edges, lying in the xy-plane. Given a prescribed orientation Θ of K, we can determine whether a translational motion for K^Θ through W exists, and, if so, find such a motion, in $O(nk \log k \log kn)$ randomized expected time.*

6 Rotations are needed

So far we have considered versions of the problem in which we were able to show that the existence of an arbitrary collision-free motion of K through W implies that K can also slide through W (or, in one instance, through another window related to W). However, perhaps not very surprisingly, this is not the case in general. We show in this and the following section that in general rotations are needed to obtain a collision-free motion of the polytope through the window.

▶ **Lemma 10.** *Let W be a square window with side length $\sqrt{5}$. Let $A = (0,0,0), B = (1,3,0), C = (1,0,h), D = (0,3,h)$ be four points, where $h \gg 1$ is a sufficiently large parameter, and let K be the tetrahedron $ABCD$ (see Figure 7). Then*

1. *K cannot pass through W by any purely translational collision-free motion (for sufficiently large $h \gg 1$).*
2. *K can pass through W by a collision-free motion with only two degrees of freedom: translating in the z-direction combined with rotation around a z-vertical axis (for any value of $h > 0$).*

The proof is given in the full version [14].

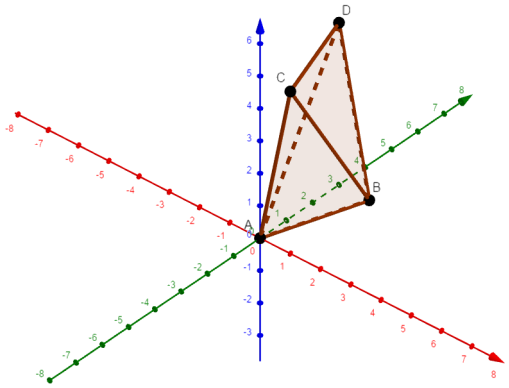


Figure 7 The tetrahedron $K = ABCD$ of Lemma 10.

7 The case of a circular window

In this section we study the case where W is a circular window. There are (at least) three possible types of motion of K through W: sliding, purely translational motion, and general motion with all six degrees of freedom. In this section we show that these types are not equivalent, as spelled out in the following theorem.

▶ **Theorem 11.** *Let K be the regular tetrahedron of side length 1. Then there exist two threshold parameters $1 > \delta_1 \approx 0.901388 > \delta_2 \approx 0.895611$, so that, denoting by d the diameter of W, we have:*

- **(i)** *K can slide through W if $d \geq 1$.*
- **(ii)** *K cannot slide through W, but can pass through W by a purely translational motion, if $\delta_1 \leq d < 1$.*
- **(iii)** *K cannot pass through W by a purely translational motion, but can pass through W by a general motion, if $\delta_2 \leq d < \delta_1$.*
- **(iv)** *K cannot pass through W at all if $d < \delta_2$.*

Proof sketch; the full proof is in [14].

(i) If $d \geq 1$ then, as is easy to show, K can slide through W, because K can be enclosed in a cylinder of diameter 1, whose axis is orthogonal to two opposite edges of K.

(ii) We need to show that, for this range of d, (a) K cannot slide through W at any fixed orientation and direction of sliding, but (b) K can pass through W by a collision-free purely translational motion.

For (a), a suitable adaptation of Lemma 1 shows that if K can slide through W in some direction then it can also slide in the vertical z-direction. It therefore suffices to show that K cannot be contained in a cylinder of diameter smaller than 1. A proof of this fact can be found in [2]. For the sake of completeness, we reproduce the proof in the full version [14], where we also provide the full details of the proof of (b).

(iii) We construct a simple five-step motion of K through W, or rather of W through K. The details are given in the full version [14], and are illustrated in Figure 8.

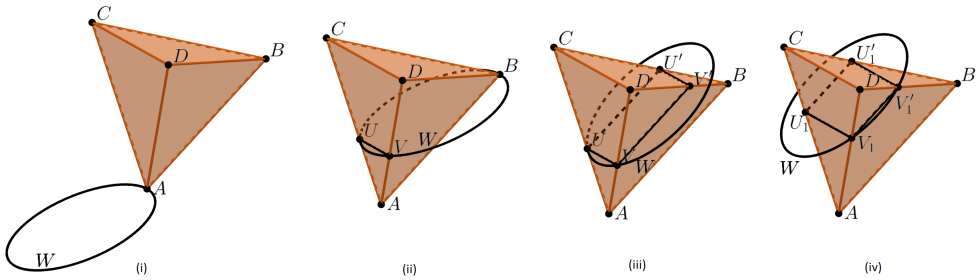


Figure 8 Moving W around K. (i) The initial configuration. (ii) Translating W until it contains the triangle BUV. (iii) Rotating W around UV until it contains the rectangle $UVV'U'$. (iv) Translating W until it contains the symmetric rectangle, with edge lengths swapped. The remainder of the motion is a fully symmetric reversal of the first two steps.

For this five-step motion to work, there has to exist a cross section through B (BUV in Figure 8(ii)) that can be enclosed by a disc of diameter d (and we show, in the full version [14], that this ensures that the entire motion is collision-free). A numerical calculation shows that the cross section through a vertex with the smallest enclosing disc is such that the diameter of this disc is $\delta_2 \approx 0.895611$. This establishes (iii).

(iv) When $d < \delta_2$ no motion is possible because no cross section through a vertex can be enclosed by a disc of diameter d, as just argued. ◀

8 Planning general rigid collision-free motion of a convex polytope through a rectangular window

Finally we deal with the general case, in which the motion of K has all six degrees of freedom. By standard (and general) arguments in algorithmic motion planning the free configuration space for this problem has complexity $O(n^6)$, and it can be computed in $O(n^{8+\varepsilon})$ time [13], from which we can easily extract a solution path, when one exists, within the same time bound. We show here that we can exploit the special structure of the problem at hand to find a solution, or detect and notify that none exists, in time close to $O(n^4)$. We sketch below the main ideas; the full details are given in the full version [14].

If there is a solution path for K to move through W with all six degrees of freedom, then there is also a canonical solution path where at all times at which K intersects the plane of W (namely the xy-plane), K touches the bottom and left edges of W with two edges e_b

and e_ℓ (possibly with the closure of these edges, namely with vertices of K, and possibly with more than one edge touching a side of W). During this motion, for every point on the path define e_t to be the edge of K whose intersection with the xy-plane has the largest y-coordinate, and e_r to be the edge of K whose intersection with the xy-plane has the largest x-coordinate.

We now split the canonical solution path into maximal open segments, along which the open edges e_b, e_ℓ, e_t, and e_r are fixed. We construct a collection of four-dimensional subspaces of the full-dimensional configuration space, one for each such quadruple e_b, e_ℓ, e_t, e_r of four edges, consisting of those free placements that have those four edges as the extreme edges in the x- and y-directions within the xy-plane. This can be done in total $O(n^4)$ time since each of these subspaces has constant descriptive complexity.

The major remaining problem is to economically detect the free connections among these $O(n^4)$ subspaces. The efficiency of our approach relies on the following lemma (proved in [14]), which asserts that the total number of certain quintuplets of edges of K that encode these connections is only $O(n^4)$, rather than $O(n^5)$, and that they can be computed efficiently:

▶ **Lemma 12.** *The maximum number of quintuplets $(e_r, e_t, e_\ell, e_b, e_\xi)$, where e_r, e_t, e_ℓ, e_b are as defined above, and e_ξ is another edge of K whose intersection with the xy-plane h_{xy} has the same x- (respectively, y-) coordinate as the intersection with h_{xy} of e_ℓ or e_r (respectively, e_b or e_t), is $O(n^4)$. All these quintuplets can be computed in $O(n^3\lambda_q(n)\log n)$ time[4] for some small constant q.*

This in turn leads to the following summary result.

▶ **Theorem 13.** *Given a convex polytope K with n edges and a rectangular window W, we can construct a collision-free motion of K through W, if one exists, or determine that no such motion exists, in time $O(n^3\lambda_q(n)\log n)$, for some small constant q. The algorithm requires $O(n^4)$ storage.*

The proofs and the algorithm are given in the full version [14].

9 Conclusion and further research

In this paper we have studied a variety of problems concerning collision-free motion of a convex polytope through a planar window, under several kinds of allowed motion – sliding (translating in a fixed direction), purely translational motion, and general motion. We have presented several structural properties and characterizations of such motions, and obtained efficient algorithms for several special cases, as well as for the general case.

There are several open problems and directions for further research. One such direction is to show that the near-quartic upper bound, established in Section 8, on the cost of the general motion planning problem for K and a rectangular window W is almost tight in the worst case, in the specific sense of establishing a lower bound $\Omega(n^4)$ on the worst-case complexity of the resulting free configuration space, a property that we conjecture to hold, and have in fact an initial plan for establishing this bound.

In addition, in Section 6, we presented an example in which a rotation is needed to pass a polytope through a rectangular window. However, in this construction we only used rotation about the line perpendicular to the plane that contains the window. This suggests

[4] $\lambda_q(n)$ is a near-linear function related to Davenport-Schinzel sequences [23].

the conjecture that every convex polytope that can pass through a rectangular window W can also pass through W by a motion consisting of arbitrary translations and rotations only about the line perpendicular to the plane of W. The results of Section 7 show that for circular windows this claim is false in general, but the status of the conjecture is still open for a rectangular window.

It is also not clear what can be said about the motion of a general non-convex polytope through a rectangular, general convex, or even non-convex window. There are several variants of this question, depending on the type of motion that we allow, both in terms of structural properties of the motion, and of the efficiency of algorithms for performing it.

References

1 `https://en.wikipedia.org/wiki/Covering_space`.

2 `https://math.stackexchange.com/questions/1364880/smallest-cylinder-into-which-a-regular-tetrahedron-can-fit`.

3 P. Bose, D. Halperin, and S. Shamai. On the separation of a polyhedron from its single-part mold. In *13th IEEE Conference on Automation Science and Engineering*, pages 61–66, 2017.

4 B. Chazelle, H. Edelsbrunner, L. J. Guibas, and M. Sharir. A singly exponential stratification scheme for real semi-algebraic varieties and its applications. *Theor. Comput. Sci.*, 84(1):77–105, 1991.

5 H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementation*. MIT Press, 2005.

6 H. T. Croft, K. J. Falconer, and R. K. Guy. *Unsolved Problems in Geometry*. Problem Books in Mathematics. Springer Verlag, Heidelberg, 1991.

7 H. E. Debrunner and P. Mani-Levitska. Can you cover your shadows? *Discrete Comput. Geom.*, 1:45–58, 1986.

8 E. Fogel, D. Halperin, and R. Wein. *CGAL Arrangements and their Applications – A Step-by-Step Guide*, volume 7 of *Geometry and Computing*. Springer, 2012.

9 P. Gibbs. A computational study of sofas and cars. *Computer Science*, 2:1–5, 2014.

10 D. Halperin, L. Kavraki, and K. Solovey. Robotics. In *Handbook of Discrete and Computational Geometry*, chapter 51, pages 1343–1376. Chapman & Hall/CRC, 3 edition, 2018.

11 D. Halperin, J.-C. Latombe, and R. H. Wilson. A general framework for assembly planning: The motion space approach. *Algorithmica*, 3-4:577–601, 2000.

12 D. Halperin, O. Salzman, and M. Sharir. Algorithmic motion planning. In *Handbook of Discrete and Computational Geometry*, chapter 50, pages 1311–1342. Chapman & Hall/CRC, 3 edition, 2018.

13 D. Halperin and M. Sharir. Arrangements. In *Handbook of Discrete and Computational Geometry*, chapter 28, pages 723–762. Chapman & Hall/CRC, 3 edition, 2018.

14 D. Halperin, M. Sharir, and I. Yehuda. Throwing a sofa through the window. *CoRR*, abs/2102.04262, 2021. `arXiv:2102.04262`.

15 A. Hatcher. *Algebraic Topology*. Cambridge University Press, Cambridge, 2002.

16 L. E. Kavraki, P. Šestka, J.-C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high dimensional configuration spaces. *IEEE Transactions on Robotics*, 12(4):566–580, 1996.

17 J. J. Kuffner and S. M. Lavalle. RRT-Connect: An efficient approach to single-query path planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 995–1001, 2000.

18 J.-C. Latombe. *Robot Motion Planning*, volume 124 of *The Kluwer international series in engineering and computer science*. Kluwer, 1991.

19 S. M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.

20 W. H. Plantinga and C. R. Dyer. Visibility, occlusion, and the aspect graph. *Int. J. Computer Vision*, 5:137–160, 1990.

21 O. Salzman, M. Hemmer, and D. Halperin. On the power of manifold samples in exploring configuration spaces and the dimensionality of narrow passages. *IEEE Trans Autom. Sci. Eng.*, 12(2):529–538, 2015.

22 O. Salzman, M. Hemmer, B. Raveh, and D. Halperin. Motion planning via manifold samples. *Algorithmica*, 67(4):547–565, 2013.

23 M. Sharir and P. K. Agarwal. *Davenport–Schinzel Sequences and Their Geometric Applications*. Cambridge University Press, New York, 1995.

24 J. Snoeyink and J. Stolfi. Objects that cannot be taken apart with two hands. *Discrete Comput. Geom.*, 12:367–384, 1994.

25 G. Toussaint. Movable separability of sets. *Comput. Geom.*, 2:335–375, 1985.

Stabbing Convex Bodies with Lines and Flats

Sariel Har-Peled
Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Il, USA

Mitchell Jones
Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL, USA

Abstract

We study the problem of constructing weak ε-nets where the stabbing elements are lines or k-flats instead of points. We study this problem in the simplest setting where it is still interesting – namely, the uniform measure of volume over the hypercube $[0,1]^d$. Specifically, a (k,ε)-net is a set of k-flats, such that any convex body in $[0,1]^d$ of volume larger than ε is stabbed by one of these k-flats. We show that for $k \geq 1$, one can construct (k,ε)-nets of size $O(1/\varepsilon^{1-k/d})$. We also prove that any such net must have size at least $\Omega(1/\varepsilon^{1-k/d})$. As a concrete example, in three dimensions all ε-heavy bodies in $[0,1]^3$ can be stabbed by $\Theta(1/\varepsilon^{2/3})$ lines. Note, that these bounds are *sublinear* in $1/\varepsilon$, and are thus somewhat surprising.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Discrete geometry, combinatorics, weak ε-nets, k-flats

Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.42

Funding *Sariel Har-Peled*: Supported in part by NSF AF award CCF-1907400.

Acknowledgements We thank an anonymous reviewer for sketching an improved construction of (k,ε)-nets for $k \geq 1$, which led to Theorem 6. Our previous construction had an additional log term.

1 Introduction

Range spaces and ε-nets. A **range space** is a pair $\mathsf{X} = (\mathcal{U}, \mathcal{R})$, where $\mathcal{U}$ is the **ground set** (finite or infinite) and $\mathcal{R}$ is a (finite or infinite) family of subsets of $\mathcal{U}$. The elements of $\mathcal{R}$ are **ranges**.

Suppose that $\mathcal{U}$ is a finite set. For a parameter $\varepsilon \in (0,1)$, a subset $\mathsf{S} \subseteq \mathcal{U}$ is an **ε-net** for the range space X, if for every range $\mathsf{r} \in \mathcal{R}$ with $|\mathsf{r} \cap \mathcal{U}| \geq \varepsilon |\mathcal{U}|$ has $\mathsf{r} \cap \mathsf{S} \neq \varnothing$. The ε-net theorem of Haussler and Welzl [5] implies the existence of ε-nets of size $O(\delta \varepsilon^{-1} \log \varepsilon^{-1})$, where δ is the VC dimension of the range space X. The use of ε-nets is widespread in computational geometry [7, 4].

Weak ε-nets. Consider the range space $(P, \mathcal{C})$, where $\mathcal{C}$ is the collection of all compact convex bodies in $\mathbb{R}^d$ and $P \subset \mathbb{R}^d$ is a point set of size n. This range space has infinite VC dimension – the standard ε-net constructions do not work for this range space. The notion of **weak ε-nets** bypasses this issue by allowing the net S to use points outside of P. Specifically, any convex body Ξ that contains at least εn points of P must contain a point of S. The first construction of weak ε-net is due to Bárány et al. [1]. There was quite a bit of work on this problem, culminating in the somewhat simpler construction of Matoušek and Wagner [8], who constructed weak ε-nets of size $O(\varepsilon^{-d} \log^{f(d)} \varepsilon^{-1})$, where $f(d) = O(d^2 \log d)$. Recently, Rubin [10, 11] gave an improved bound, showing existence of weak ε-nets of size $O(\varepsilon^{-(d-0.5+\alpha)})$ for arbitrarily small $\alpha > 0$. For more detailed history of the problem, see the introduction of Rubin [10, 11]. As for a lower bound, Bukh et al. [2] gave constructions of point sets for which any weak ε-net must have size $\Omega(\varepsilon^{-1} \log^{d-1} \varepsilon^{-1})$. Closing this gap remains a major open problem.

(k, ε)-nets and uniform measure. A natural extension of weak ε-nets is to allow the net S to contain other geometric objects. Given a collection of n points $P \subset \mathbb{R}^d$ and a parameter $0 \leq k < d$, we define a (weak) (k, ε)-net to be a collection of k-flats S such that if Ξ is a convex body containing at least εn points of P, then there exists a k-flat in S intersecting Ξ. Note that $(0, \varepsilon)$-nets are exactly weak ε-nets.

In general, one would expect that as k increases, the size of the (k, ε)-net shrinks. For example, a $(1, \varepsilon)$-net for a collection of points in $\mathbb{R}^3$ can be constructed by projecting the points down onto the xy-plane and applying Rubin's construction in the plane to obtain a weak ε-net S of size $O(\varepsilon^{-(3/2+\alpha)})$ [10]. Lifting S up back into three dimensions results in a $(1, \varepsilon)$-net of the same size, which is smaller than the best known weak ε-net size in $\mathbb{R}^3$ [8, 10, 11]. However, one might expect that a $(1, \varepsilon)$-net of even smaller size is possible in $\mathbb{R}^3$, as this construction uses a set of parallel lines (i.e., one would expect the lines in an optimal net to be arbitrarily oriented).

Here, we study an even simpler version of the problem, where the ground set is the hypercube $B = [0, 1]^d$. In particular, for $\varepsilon \in (0, 1)$ and $0 \leq k < d$, we are interested in computing the smallest set K of k-flats, such that if Ξ is a convex body with $\mathsf{vol}(\Xi \cap B) \geq \varepsilon$, then there is a k-flat in K which intersects Ξ. For sake of exposition, throughout the rest of the paper we refer to this set K as a **(k, ε)-net for volume measure**. We note that $[0, 1]^d$ can be replaced with any arbitrary compact convex body in the definition (the size of the (k, ε)-net increases by a factor depending on d, see Appendix B).

1.1 Our results & paper organization

Notation. Throughout, the notation O_d, Ω_d, and Θ_d hides constants depending on the dimension d.

First, we show that any (k, ε)-net for volume measure must have size $\Omega_d(1/\varepsilon^{1-k/d})$ (Lemma 3). Perhaps surprisingly, we give a relatively simple construction of (k, ε)-nets for volume measure of size $O_d(1/\varepsilon^{1-k/d})$ for $k \geq 1$ (Theorem 6). For $k = 0$, we obtain nets of size $O_d((1/\varepsilon)\log^{d-1}(1/\varepsilon))$ (Theorem 11). Importantly, both constructions are deterministic and explicit (see the discussion below).

As far as the authors are aware, this particular problem we study has not been addressed before. The only related result known is the existence of explicit constructions of $(0, \varepsilon)$-nets for volume measure for axis parallel boxes in $\mathbb{R}^d$, and is briefly mentioned in [2]. In this case, one can construct a $(0, \varepsilon)$-net for volume measure of size $O_d(1/\varepsilon)$ using Van der Corput sets in two dimensions, and Halton-Hammersely sets in higher dimensions. For completeness, we describe these construction in Appendix A.

Deterministic vs. explicit constructions of ε-nets. For the regular concept of ε-nets, there are known deterministic constructions. They work by repeatedly halving the input point set, using deterministic discrepancy constructions, until the set is of the desired size [6, 3]. On the one hand, for our setting (i.e., the measure is uniform volume on the unit hypercube) it is not clear what the generated ε-net is without running this construction algorithm outright. On the other hand, we develop a construction of weak ε-nets – for uniform volume measure over the hypercube for ellipsoids – which are much simpler and are explicit; one can easily compute the ith point in this net using polylogarithmic space.

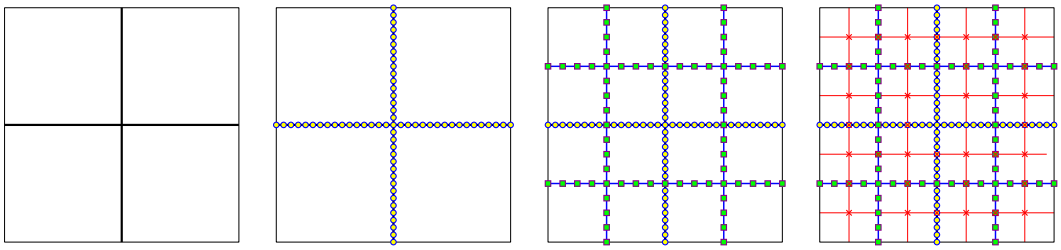


Figure 3.1 The multi-level grid, and its associated lines.

2 Lower bound

▶ **Definition 1.** *The affine hull of a point set $P = \{p_1, \ldots, p_n\} \subseteq \mathbb{R}^d$ is the set*

$$\left\{ \sum_i \alpha_i p_i \;\middle|\; \forall i \quad \alpha_i \in \mathbb{R} \quad \text{and} \quad \sum_i \alpha_i = 1 \right\}.$$

*For $0 \leq k < d$, a **k-flat** is the affine hull of a set of $k+1$ (affinely independent) points.*

▶ **Definition 2.** *For parameters $\varepsilon \in (0,1)$ and $k \in \{0, 1, \ldots, d-1\}$, a set K of k-flats is a **(k,ε)-net for volume measure** if for any convex body $\Xi \subseteq \mathbb{R}^d$ with $\mathsf{vol}(\Xi \cap [0,1]^d) \geq \varepsilon$, there exists a flat $\varphi \in K$ such that $\varphi \cap \Xi \neq \emptyset$.*

▶ **Lemma 3.** *For a parameter $\varepsilon \in (0,1)$, any (k,ε)-net for volume measure must have size $\Omega_d(1/\varepsilon^{1-k/d})$.*

Proof. Let K be a (k,ε)-net for volume measure. For each k-flat $\varphi \in K$, let $H(\varphi, r)$ be the locus of points in $[0,1]^d$ within distance at most r from φ (for $k = 1$ in three dimensions, this is the intersection of $[0,1]^d$ and the cylinder with radius r centered at the line φ). Note that a ball b with center c and radius r intersects a k-flat φ if and only if $c \in H(\varphi, r)$.

Fix $r = (\varepsilon/\mu)^{1/d}$, where μ is a constant to be determined shortly. We claim that by choosing μ appropriately, if K is a (k,ε)-net for volume measure, then the collection of objects $\{H(\varphi, r) \mid \varphi \in K\}$ covers $[0,1]^d$. Indeed, suppose not. Then there exists a point $p \in [0,1]^d$ not covered by any of the objects $H(\varphi, r)$. This implies that a ball b centered at p with radius r does not intersect any k-flat of K, and its volume is $c_d r^d = c_d \varepsilon/\mu$, where c_d is a constant that depends on d. Choose $\mu = c_d$ so that b has volume at least ε, but does intersect any k-flat of K. A contradiction to the required net property.

Hence, by the choice of r, any (k,ε)-net for volume measure must satisfy the condition that $\{H(\varphi, r) \mid \varphi \in K\}$ covers $[0,1]^d$. For any k-flat φ, we have $\beta = \mathsf{vol}(H(\varphi, r)) = O_d(r^{d-k}) = O_d(\varepsilon^{1-k/d})$. Thus, to cover $[0,1]^d$, we have that $|K| \geq 1/\beta = \Omega_d(1/\varepsilon^{1-k/d})$. ◀

3 Constructing (k,ε)-nets for volume measure for $k \geq 1$

Here, we give a self-contained proof of a deterministic, explicit construction of (k,ε)-nets for volume measure of size $O_d(1/\varepsilon^{1-k/d})$ for $k \geq 1$ which matches the lower bound of Lemma 3 up to constant factors. The construction will be done recursively on the dimension d.

Base case: $k = d-1$. Here a $(d-1,\varepsilon)$-net for volume measure of size $d/\varepsilon^{1/d} = O_d(1/\varepsilon^{1-k/d})$ follows readily by overlaying a d-dimensional grid of size length $\varepsilon^{1/d}$ and letting the net consist of the hyperplanes forming the grid. As such, we assume $k < d-1$.

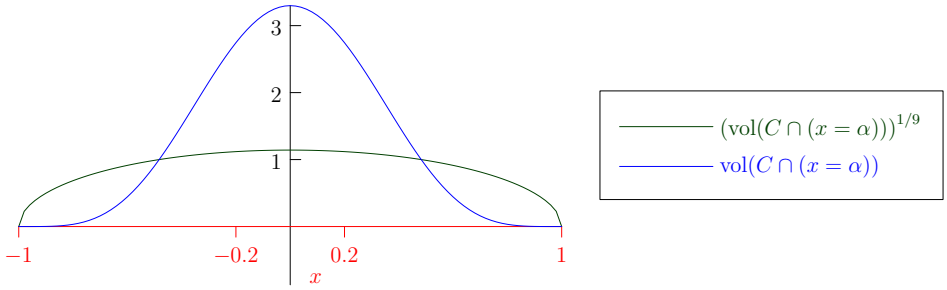


Figure 3.2 The slice volume, and its 1/9th power, for the unit radius ball Ξ in 10 dimensions. This is an example of the concavity implied by the Brunn-Minkowski inequality, which in turn implies that the slice function is unimodal.

3.1 Construction

The construction is based on quadtrees. Starting with the entire cube $[0,1]^d$, we construct d orthogonal hyperplanes which *split* the cube into 2^d cubes of side length $1/2$. We refer to such hyperplanes as **splitting hyperplanes**. This splitting process is continued recursively inside each cell, for $i = 0, \ldots, \tau$, where

$$\tau = \left\lceil \frac{1}{d} \lg \frac{1}{\varepsilon} \right\rceil + 3 \lceil \log(3d) \rceil + 1 \tag{3.1}$$

(and $\lg = \log_2$), so that cubes at the ith level of the construction has side length $1/2^i$. The number of such cubes at the ith level is 2^{di}. Naturally, these cubes together form a grid with side length $1/2^i$. See Figure 3.1 for an illustration of the construction in two dimensions.

For each splitting hyperplane h at level $i \geq 1$, which splits cells of side length $1/2^{i-1}$ into cells of side length $1/2^i$, we recursively construct a (k, ε_i)-net for volume measure on h (which lies in $d-1$ dimensions), where

$$\varepsilon_i = \frac{2^i \varepsilon}{4d}. \tag{3.2}$$

We collect all k-flats on all splitting hyperplanes at all levels into our (k,ε)-net for volume measure K.

3.2 Analysis

▶ **Lemma 4.** *The constructed (k,ε)-net for volume measure has size $O_d(1/\varepsilon^{1-k/d})$.*

Proof. Let $T(\varepsilon, d)$ denote the minimum size of a (k,ε)-net for volume measure over $[0,1]^d$. The proof is by induction on d. When $d = k+1$, we have $T(\varepsilon, k+1) \leq (k+1)/\varepsilon^{1/(k+1)}$, by the base case described above. So assume $d \geq k+2$ and $T(\delta, d') \leq \beta(d')/\delta^{1-k/d'}$ for all $d' < d$, where $\beta(d')$ is a constant to be determined. By the inductive hypothesis, the above construction produces a (k,ε)-net for volume measure of size

$$\begin{aligned} |K| &\leq d \sum_{i=1}^{\tau} 2^{i-1} T(\varepsilon_i, d-1) \leq d \sum_{i=1}^{\tau} \frac{2^{i-1}\beta(d-1)}{\varepsilon_i^{1-k/(d-1)}} \leq \frac{4d^2\beta(d-1)}{\varepsilon^{1-k/d}} \sum_{i=1}^{\tau} \frac{2^{i-1}}{2^{i-ik/(d-1)}} \\ &\leq \frac{2d^2\beta(d-1)}{\varepsilon^{1-k/d}} \sum_{i=1}^{\tau} 2^{ik/(d-1)} \leq \frac{4d^2\beta(d-1)}{\varepsilon^{1-k/(d-1)}} \cdot 2^{\tau k/(d-1)} \leq \frac{16d^2\beta(d-1)}{\varepsilon^{1-k/d}}. \end{aligned}$$

The last inequality follows since $\tau \leq \frac{1}{d}\lg\frac{1}{\varepsilon} + 2$. In particular, we obtain the recurrence $\beta(d) = 16d^2\beta(d-1)$, which solves to $\beta(d) = d^{O(d)}$. As such, the size of K is $O_d(1/\varepsilon^{1-k/d})$. ◀

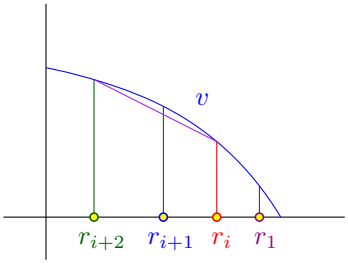


Figure 3.3 By the choice of $\boldsymbol{r}_\tau \leq \ldots \leq \boldsymbol{r}_1$, we have $v(\boldsymbol{r}_\tau) \geq \ldots \geq v(\boldsymbol{r}_1)$.

The Brunn-Minkowski inequality and unimodal functions. The Ξ be a convex body in $\mathbb{R}^d$. For a parameter $\alpha \in \mathbb{R}$, let $f(\alpha)$ denote the $(d-1)$-dimensional volume of Ξ intersected with the hyperplane $x = \alpha$. The Brunn-Minkowski inequality [7, 4] implies that the function $g(\alpha) = f(\alpha)^{1/(d-1)}$ is concave. In particular, g is **unimodal**. Namely, there exists a $\alpha \in \mathbb{R}$ such that g is non-decreasing on $(-\infty, \alpha]$ and non-increasing on $[\alpha, \infty)$. As such, the function f itself is unimodal. See Figure 3.2.

▶ **Lemma 5.** *The set K is a (k, ε)-net for volume measure.*

Proof. Let Ξ be a convex body contained in $[0,1]^d$ with volume at least ε. Assume, for the sake of contradiction, that Ξ is not stabbed by any of the k-flats of K.

Let $h(\alpha)$ be the hyperplane orthogonal to the first axis which intersects the first axis at $\alpha \in \mathbb{R}$. Define the function

$$f(\alpha) = \mathsf{vol}\big(\Xi \cap h(\alpha)\big).$$

By the Brunn-Minkowski inequality, the function $g(\alpha) = f(\alpha)^{1/(d-1)}$ is concave and unimodal. Define the point $x^* \in [0,1]$ so that $x^\star = \arg\max_\alpha f(\alpha)$.

Let $V(\Delta) = f(x^\star + \Delta)$, and let $v(\Delta) = (V(\Delta))^{1/(d-1)}$. The function v, being a translation of g, is concave and unimodal. Let $\boldsymbol{r}_i \geq 0$ be the maximum number such that $V(\boldsymbol{r}_i) = \varepsilon_i$, for $i = 1, \ldots, \tau$. Observe that if $\boldsymbol{r}_i \geq 1/2^i$, then there is hyperplane orthogonal to the first axis that has a recursive construction of a net on it, for ε_i. This by induction would imply that the net intersects Ξ. We thus assume from this point on that

$$\boldsymbol{r}_i < \frac{1}{2^i},$$

for all i. Observe that $\boldsymbol{r}_1 \geq \boldsymbol{r}_2 \geq \cdots \geq \boldsymbol{r}_\tau$, as $\varepsilon_1 < \varepsilon_2 < \cdots < \varepsilon_\tau$ (more specifically, $\varepsilon_i = 2\varepsilon_{i-1}$ for all i).

The concavity of $v(\cdot)$, see Figure 3.3, implies that

$$\frac{v(\boldsymbol{r}_{i+2}) - v(\boldsymbol{r}_{i+1})}{\boldsymbol{r}_{i+2} - \boldsymbol{r}_{i+1}} \geq \frac{v(\boldsymbol{r}_{i+1}) - v(\boldsymbol{r}_i)}{\boldsymbol{r}_{i+1} - \boldsymbol{r}_i} \qquad \Longrightarrow \qquad \frac{\boldsymbol{r}_{i+1} - \boldsymbol{r}_i}{\boldsymbol{r}_{i+2} - \boldsymbol{r}_{i+1}} \leq \frac{v(\boldsymbol{r}_{i+1}) - v(\boldsymbol{r}_i)}{v(\boldsymbol{r}_{i+2}) - v(\boldsymbol{r}_{i+1})},$$

as $\boldsymbol{r}_{i+1} - \boldsymbol{r}_i < 0$ and $v(\boldsymbol{r}_{i+2}) - v(\boldsymbol{r}_{i+1}) > 0$. Since $V(\boldsymbol{r}_{i+1}) = \varepsilon_{i+1} = 2\varepsilon_i = 2V(\boldsymbol{r}_i)$, we have that $v(\boldsymbol{r}_{i+1}) = 2^{1/(d-1)} v(\boldsymbol{r}_i)$. For $i < \tau$, let $\ell_i = \boldsymbol{r}_i - \boldsymbol{r}_{i+1}$. Plugging this into the above, observe

$$\frac{\ell_i}{\ell_{i+1}} = \frac{\boldsymbol{r}_i - \boldsymbol{r}_{i+1}}{\boldsymbol{r}_{i+1} - \boldsymbol{r}_{i+2}} \leq \frac{v(\boldsymbol{r}_{i+1}) - v(\boldsymbol{r}_i)}{v(\boldsymbol{r}_{i+2}) - v(\boldsymbol{r}_{i+1})} = \frac{(2^{1/(d-1)} - 1) v(\boldsymbol{r}_i)}{2^{1/(d-1)}(2^{1/(d-1)} - 1) v(\boldsymbol{r}_i)} = \frac{1}{2^{1/(d-1)}}.$$

Since $\ell_{\tau-1} \leq r_{\tau-1} \leq 1/2^{\tau-1}$, we have

$$r_1 = r_\tau + \sum_{i=1}^{\tau-1} \ell_i \leq r_\tau + \ell_{\tau-1}\left(1 + \frac{1}{2^{1/(d-1)}} + \frac{1}{2^{2/(d-1)}} + \cdots\right)$$
$$\leq r_\tau + 2d\ell_{\tau-1} \leq (2d+1)r_{\tau-1} < \frac{2d+1}{2^{\tau-1}} < \frac{\varepsilon^{1/d}}{4d^2},$$

by the value of τ, see Eq. (3.1).

Let I_1 be the maximum interval, where the value of $V(x) \geq \varepsilon_1$, for any $x \in I_1$. By the above, we have that if the net does not intersect Ξ, then $\|I_1\| \leq 2r_1 \leq 2\varepsilon^{1/d}/(4d^2)$.

We define $I_2, \ldots, I_d$ in a similar fashion on the other axes, and the same argumentation would imply that $\|I_j\| \leq 2\varepsilon^{1/d}/(4d^2)$, for all j. Furthermore, any plane orthogonal to the axes that avoids the box $B = I_1 \times I_2 \cdots \times I_d$ has an intersection with Ξ of volume at most ε_1. We conclude that the total value of Ξ is at most

$$\mathsf{vol}(\Xi) \leq \mathsf{vol}(B) + \sum_{j=1}^{d} \int_{y \in [0,1] \setminus I_j} \mathsf{vol}\Big(\Xi \cap (x_j = y)\Big) dy \leq \prod_{j=1}^{d} \|I_j\| + d\varepsilon_1 \ll \varepsilon,$$

which is a contradiction to $\mathsf{vol}(\Xi) \geq \varepsilon$. ◀

▶ **Theorem 6.** *Given $\varepsilon \in (0,1)$ and $k \in \{1, \ldots, d-1\}$, the above is a deterministic and explicit construction of a (k, ε)-net for volume measure over $[0,1]^d$ of size $O_d(1/\varepsilon^{1-k/d})$.*

4 Constructing $(0, \varepsilon)$-nets for volume measure

4.1 Ellipsoids are enough

We now give constructions for $(0, \varepsilon)$-nets for volume measure. The following result shows that it suffices to build such nets when the convex bodies are restricted to be ellipsoids.

▶ **Lemma 7.** *Suppose there exists a $(0, \varepsilon)$-net for volume measure over $[0,1]^d$ for ellipsoids of size $T(\varepsilon, \tau)$, for $\tau = 1, \ldots, d$. Then one can construct a $(0, \varepsilon)$-net for volume measure over $[0,1]^d$ of size $T(\varepsilon/d^d, d)$.*

Proof. Consider any convex body Ξ, such that $\mathsf{vol}\big(\Xi \cap [0,1]^d\big) \geq \varepsilon$. Let $\mathcal{E}$ be the ellipsoid of largest volume contained inside $\Xi \cap [0,1]^d$. By John's ellipsoid theorem, we have that $\mathcal{E} \subseteq \Xi \subseteq d\mathcal{E}$. In particular,

$$\mathsf{vol}(\mathcal{E}) = \mathsf{vol}(d\mathcal{E})/d^d \geq \frac{\mathsf{vol}(\Xi)}{d^d} \geq \frac{\varepsilon}{d^d}.$$

As such, any $(0, \varepsilon/d^d)$-net for volume measure when the convex bodies are restricted to be ellipsoids is a $(0, \varepsilon)$-net for volume measure in the general setting. ◀

Hence, we focus on building $(0, \varepsilon)$-nets for volume measure (equivalently, these are also ε-nets for volume measure) for ellipsoids. Note that it is easy to obtain an ε-net of size $O_d(\varepsilon^{-1} \log \varepsilon^{-1})$ by random sampling [5]. Here, we give a deterministic, explicit construction of such a net.

4.2 Stabbing ellipsoids with points

4.2.1 Net construction in 2D

Let $\mathcal{E}$ be an ellipse contained in the unit square $[0,1]^2$ with $\mathsf{area}(\mathcal{E}) \geq \varepsilon$. The following construction is inspired by a construction of Pach and Tardos [9].

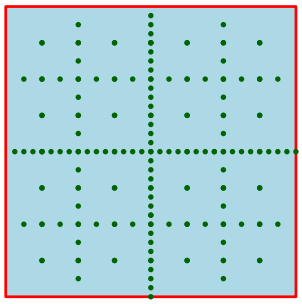


Figure 4.1 The net constructed.

Construction. Let $M = 3 + \lceil \lg \varepsilon^{-1} \rceil$. For $j = 1, \ldots, M-1$, consider the rectangle

$$R_j = [0, 1/2^{M-j}] \times [0, 1/2^j].$$

Consider the natural tiling of $[0,1]^2$ by the rectangle R_i, and let P_i be the set of vertices of the resulting grid G_i in the interior of the unit square. Let $N = \cup_i P_i$. See Figure 4.1.

Correctness. We need the following easy observation, whose proof is included for the sake of completeness.

▷ Claim 8. Let c be the center of an ellipse $\mathcal{E}$, and let $\hbar$ be the longest horizontal segment contained in $\mathcal{E}$. The segment $\hbar$ passes through c.

Proof. By the central symmetry of $\mathcal{E}$, if $\hbar$ does not pass through c, then it has a symmetric reflection $\hbar'$ through c, which is a horizontal segment of the same length. Let ℓ be the horizontal line through c, and observe that $|\ell \cap \mathcal{E}| \geq |\hbar|$ by convexity. By the smoothness of $\mathcal{E}$, it follows that $|\ell \cap \mathcal{E}| > |\hbar|$, which is a contradiction. ◁

▶ **Lemma 9.** *The set N constructed above is an ε-net for volume measure over $[0,1]^2$ for ellipses. Furthermore, $|N| = O(\varepsilon^{-1} \log \varepsilon^{-1})$.*

Proof. Observe that for any i, we have $\mathsf{area}(R_i) = 2^{-(M-j)-j} = 2^{-M} \geq \varepsilon/8$. As such, $|P_i| = O(1/\varepsilon)$, and $|N| = O(M/\varepsilon) = O(\varepsilon^{-1} \log \varepsilon^{-1})$.

Let $\mathcal{E} \subseteq [0,1]^2$ be any ellipse with $\mathsf{area}(\mathcal{E}) \geq \varepsilon$. Let Y denote the projection of $\mathcal{E}$ onto the y-axis. Observe that $|Y| \geq \varepsilon$. Let $\hbar$ be the longest horizontal segment contained in $\mathcal{E}$ (which passes through the center of $\mathcal{E}$ by Claim 8). The two extreme y-axis points in $\mathcal{E}$, and the segment $\hbar$ forms a quadrilateral in $\mathcal{E}$ of area $|\hbar|\,|Y|/2$, see Figure 4.2. Let $Y = [y_-, y_+]$, and for $\alpha \in Y$, let $g(\alpha) = |\{y = \alpha\} \cap \mathcal{E}|$. We have that

$$|\hbar|\,|Y|/2 \leq \mathsf{area}(\mathcal{E}) = \int_{\alpha=y_-}^{y_+} g(\alpha) \mathrm{d}\alpha \leq |\hbar|\,|Y|\,.$$

Since $\mathsf{area}(\mathcal{E}) \geq \varepsilon$, we conclude that $|\hbar| \geq \varepsilon/|Y|$.

We set $y_{1/4} = (3/4)y_- + (1/4)y_+$ and $y_{3/4} = (1/4)y_- + (3/4)y_+$. Consider the two horizontal segments $\hbar_{1/4} = \{y = y_{1/4}\} \cap \mathcal{E}$ and $\hbar_{3/4} = \{y = y_{3/4}\} \cap \mathcal{E}$. These two segments are of the same length and are parallel. Furthermore, $\gamma = |\hbar_{1/4}| = |\hbar_{3/4}| \geq |\hbar|/2$, see Figure 4.2. Consider the parallelogram Z formed by the convex hull of $\hbar_{1/4}$ and $\hbar_{3/4}$. Observe, that for any $\alpha \in [y_{1/4}, y_{3/4}]$, we have that $|\{y = \alpha\} \cap Z| = \gamma$. As such, $\mathsf{area}(Z) = \gamma \cdot |Y|/2 \geq |\hbar|/2 \cdot |Y|/2 \geq \varepsilon/4$. Let k be the minimum integer such that $1/2^{k+1} \leq |Y|/2$. Since $|Y| \geq \varepsilon$, it follows that $k < M - 2$.

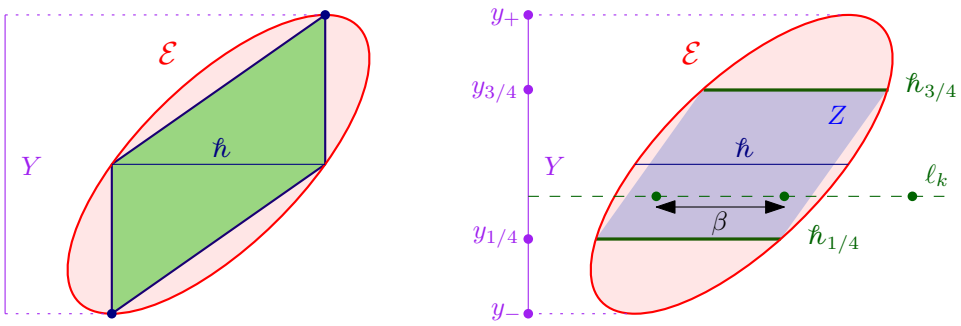


Figure 4.2 The setup for proof of correctness.

This implies that the grid G_{k+1} has a horizontal line ℓ_k that intersects Z. Furthermore, we have

$$|\ell_k \cap \mathcal{E}| \geq |\ell_k \cap Z| = \gamma \geq \frac{|\hbar|}{2} \geq \frac{\varepsilon}{2|Y|} \geq \varepsilon 2^k \geq \frac{8 \cdot 2^k}{2^M} = \frac{1}{2^{M-k-3}} > \frac{1}{2^{M-(k+1)}} = \beta,$$

since $M = 3 + \lceil \lg \varepsilon^{-1} \rceil$. Namely, the spacing of the points of G_{k+1} on the line ℓ_k (i.e., β) is shorter than the interval $\ell_k \cap \mathcal{E}$. It follows that a point of $P_{k+1} \subseteq N$ lies in $\mathcal{E}$, and thus establishing the claim. ◀

4.2.2 The construction in higher dimensions

We now extend the previous construction to higher dimensions. The construction is recursive. Namely, we assume that for all $d' < d$ we can construct an ε-net for volume measure over $[0,1]^{d'}$ for ellipsoids of size $(\beta(d')/\varepsilon) \lg^{d'-1}(1/\varepsilon)$, where $\beta(d')$ is a constant depending on the dimension d' (to be determined shortly). Lemma 9 proves the claim when $d = 2$.

Construction. Label the d axes $x_1, \ldots, x_d$. Let $\tau = \lceil (1/d) \lg(1/\varepsilon) \rceil$ and define the function $\Delta(i) = 2^i \varepsilon^{1/d}$. We repeat the following construction for each axis x_ℓ, where $\ell = 1, \ldots, d$. For each $i = 0, \ldots, \tau$, let $M_i = \lceil \lg(1/\Delta(i)) \rceil$. For each i, and for each $j = 0, \ldots, M_i$, form $2^j + 1$ evenly spaced hyperplanes which are orthogonal to the axis x_ℓ (thus consecutive hyperplanes are separated by distance 2^{-j}). For each hyperplane h, we recursively construct a $\varepsilon/\Delta(i+2)$-net $P_{\ell,i,j}$ for $[0,1]^{d-1}$ on $h \cap [0,1]^d$. Let $P_\ell = \cup_{i=1}^{\tau} \cup_{j=1}^{M_i} P_{\ell,i,j}$. Finally, we claim the point set $P = \cup_{\ell=1}^{d} P_\ell$ is the desired ε-net for volume measure.

▶ **Theorem 10.** *For $\varepsilon \in (0, 2^{-2d}]$, there exists a ε-net for volume measure over $[0,1]^d$ for ellipsoids, of size $2^{O(d^2)} \varepsilon^{-1} \lg^{d-1} \varepsilon^{-1}$.*

Proof. We first bound the size of the resulting net. Since $\varepsilon \leq 2^{-2d}$, by a direct calculation,

$$\begin{aligned}
|P| &\leq \sum_{\ell=1}^{d} |P_\ell| \leq d \sum_{i=0}^{\tau} \sum_{j=0}^{M_i} (2^j + 1) \cdot \beta(d-1) \cdot \left(\frac{\Delta(i+2)}{\varepsilon} \lg^{d-2} \left(\frac{\Delta(i+2)}{\varepsilon} \right) \right) \\
&\leq \frac{2d \cdot \beta(d-1)}{\varepsilon} \sum_{i=0}^{\tau} 2^{M_i+1} \cdot 2^2 \Delta(i) \lg^{d-2} \left(\frac{\Delta(i+2)}{\varepsilon} \right) \\
&\leq \frac{2^5 d \cdot \beta(d-1)}{\varepsilon} \sum_{i=0}^{\tau} \lg^{d-2} \left(\frac{2^{i+2}}{\varepsilon^{1-1/d}} \right)
\end{aligned}$$

$$\leq \frac{2^5 d \cdot \beta(d-1)}{\varepsilon} \sum_{i=0}^{\tau} \left((i+2) + \lg\left(\frac{1}{\varepsilon^{1-1/d}}\right)\right)^{d-2}.$$

Since $i+2 \leq \tau+2 \leq \lg(1/\varepsilon)$ for $\varepsilon \leq 2^{-2d}$, we have

$$|P| \leq \frac{2^5 d \cdot \beta(d-1)}{\varepsilon} \left[(\tau+1) \cdot 2^{d-2} \lg^{d-2}\left(\frac{1}{\varepsilon}\right)\right] \leq \frac{2^5 d \cdot \beta(d-1)}{\varepsilon} \left[\frac{4}{d} \lg \frac{1}{\varepsilon} \cdot 2^{d-2} \lg^{d-2} \frac{1}{\varepsilon}\right].$$

As such, $|P| \leq \frac{2^{d+5} \cdot \beta(d-1)}{\varepsilon} \lg^{d-1}\left(\frac{1}{\varepsilon}\right)$. In particular, we obtain the recurrence $\beta(d) = 2^{d+5}\beta(d-1)$, which solves to $\beta(d) = 2^{O(d^2)}$. Hence, $|P| = 2^{O(d^2)} \varepsilon^{-1} \lg^{d-1} \varepsilon^{-1}$.

We now argue correctness. Let $\mathcal{E}$ be an ellipsoid of volume at least ε. Let B be the smallest enclosing axis-aligned box for $\mathcal{E}$. Suppose that the longest edge of B is along the ℓth axis. In particular, along this ℓth axis B has side length $s \geq \varepsilon^{1/d}$, for otherwise $\mathsf{vol}(\mathcal{E}) \leq \mathsf{vol}(B) \leq s^d < \varepsilon$. We claim that $\mathcal{E}$ intersects a point in the set P_ℓ.

Let $L = [\ell_-, \ell_+]$ be the projection of $\mathcal{E}$ onto the ℓth axis, with $s = |L|$. For $x \in L$, define $H(x)$ to be the hyperplane orthogonal to the ℓth axis which intersects the ℓth axis at x. Finally, let K be the hyperplane through the center of $\mathcal{E}$ which is orthogonal to the ℓth axis and set $\mathcal{F} = \mathcal{E} \cap K$. We claim that $\mathsf{vol}(\mathcal{F}) \geq \varepsilon/s$. To prove the claim, suppose towards contradiction that $\mathsf{vol}(\mathcal{E} \cap K) < \varepsilon/s$. Then,

$$\mathsf{vol}(\mathcal{E}) = \int_{\ell_-}^{\ell_+} \mathsf{vol}(\mathcal{E} \cap H(x))\,\mathrm{d}x < \frac{\varepsilon}{s} \int_{\ell_-}^{\ell_+} 1\,\mathrm{d}x = \frac{\varepsilon}{s}|L| = \varepsilon,$$

a contradiction.

Choose an integer $i \geq 0$ such that $s \in [\Delta(i), \Delta(i+1))$. Let $z_{1/4} = (3/4)\ell_- + (1/4)\ell_+$ and $z_{3/4} = (1/4)\ell_- + (3/4)\ell_+$. Observe that for all $x \in [z_{1/4}, z_{3/4}]$, $\mathsf{vol}(\mathcal{E} \cap H(x)) \geq \varepsilon/(2s) \geq \varepsilon/\Delta(i+2)$. Next, let j be the minimum integer such that $1/2^{j+1} \leq s/2$. Note that such an integer exists, as we can choose $j = \lceil \lg(1/s) \rceil$. Since $s \geq \Delta(i)$, $j \leq \lceil \lg(1/\Delta(i)) \rceil \leq M_i$. Thus, for our choices of i and j, we have found a hyperplane h which intersects $\mathcal{E}$ with $\mathsf{vol}(\mathcal{E} \cap h) \geq \varepsilon/\Delta(i+2)$. By our recursive construction, there is a point in the net $P_{\ell,i,j}$ which intersects $\mathcal{E} \cap h$ and thus $\mathcal{E}$. ◀

▶ **Theorem 11.** *There is a deterministic, explicit construction of $(0,\varepsilon)$-nets for volume measure over $[0,1]^d$ of size*

$$O_d\left(\frac{1}{\varepsilon} \log^{d-1} \frac{1}{\varepsilon}\right).$$

Proof. Follows by plugging in the bound for Theorem 10 into Lemma 7. ◀

5 Conclusion

The main open problem left by our work is bounding the size of (k,ε)-nets in the general case. That is, the input is a set P of n points in $\mathbb{R}^d$, and we would like to compute a minimum set of k-flats which stab all convex bodies containing at least εn points of P. As noted earlier, there is a (k,ε)-net of asymptotically the same size as of a weak ε-net in $\mathbb{R}^{d-k}$. This follows by projecting the point set to a subspace of dimension $d-k$, constructing a regular weak ε-net, and lifting the net back to the original space. Can one do better than this somewhat naive construction?

Note that it is easy to show a lower bound of size $\Omega(1/\varepsilon)$ for $(1,\varepsilon)$-nets in the general case. Take a point set that consists of $\lceil 2/\varepsilon \rceil$ equally sized clusters of tightly packed points, such that no line passes through three clusters. Namely, our sublinear results in $1/\varepsilon$ are special for the uniform measure on the hypercube.

References

1 Imre Bárány, Zoltán Füredi, and László Lovász. On the number of halving planes. *Combinatorica*, 10:175–183, 1990. doi:10.1007/BF02123008.

2 Boris Bukh, Jiří Matoušek, and Gabriel Nivasch. Lower bounds for weak epsilon-nets and stair-convexity. In *Proc. 25th Annu. Sympos. Comput. Geom.* (SoCG), pages 1–10. ACM, 2009. doi:10.1145/1542362.1542365.

3 Bernard Chazelle. *The Discrepancy Method: Randomness and Complexity*. Cambridge University Press, New York, 2001. URL: http://www.cs.princeton.edu/~chazelle/book.html.

4 Sariel Har-Peled. *Geometric Approximation Algorithms*, volume 173 of *Math. Surveys & Monographs*. Amer. Math. Soc., Boston, MA, USA, 2011. doi:10.1090/surv/173.

5 David Haussler and Emo Welzl. ε-nets and simplex range queries. *Discrete Comput. Geom.*, 2:127–151, 1987. doi:10.1007/BF02187876.

6 Jiří Matoušek. *Geometric discrepancy: An illustrated guide*, volume 18 of *Algorithms and Combinatorics*. Springer, 1999. doi:10.1007/978-3-642-03942-3.

7 Jiří Matoušek. *Lectures on Discrete Geometry*, volume 212 of *Grad. Text in Math.* Springer, 2002. doi:10.1007/978-1-4613-0039-7.

8 Jiří Matoušek and Uli Wagner. New constructions of weak ε-nets. *Discrete Comput. Geom.*, 32(2):195–206, 2004. doi:10.1007/s00454-004-1116-4.

9 János Pach and Gábor Tardos. Tight lower bounds for the size of epsilon-nets. *J. Amer. Math. Soc.*, 26:645–658, 2013. doi:10.1090/S0894-0347-2012-00759-0.

10 Natan Rubin. An improved bound for weak epsilon-nets in the plane. In *Proc. 59th Annu. IEEE Sympos. Found. Comput. Sci.* (FOCS), pages 224–235, 2018. doi:10.1109/FOCS.2018.00030.

11 Natan Rubin. Stronger bounds for weak epsilon-nets in higher dimensions. In *Proc. 53rd ACM Sympos. Theory Comput.* (STOC), 2021.

A $(0, \varepsilon)$-nets for volume measure when the bodies are axis-aligned boxes

Here we show the existence of a $(0, \varepsilon)$-net for volume measure of size $O(1/\varepsilon)$ that intersects any axis-aligned box B with $\mathsf{vol}\big(B \cap [0,1]^2\big) \geq \varepsilon$. The following constructions are essentially described in [6] (in the context of low-discrepancy point sets), however the proofs use similar tools. We give the proofs for completeness.

▶ **Definition 12** (the Van der Corput set)**.** *For an integer α, let $\mathsf{bin}(\alpha) \in \{0,1\}^\star$ denote the binary representation of α, and $\mathsf{rev}(\mathsf{bin}(\alpha))$ be the reversal of the string of digits in $\mathsf{bin}(\alpha)$. We define $\mathsf{br}(\alpha) \in [0,1]$ to be the **bit-reversal** of α, which is defined as the number obtained by concatenating "0." with the string $\mathsf{rev}(\mathsf{bin}(\alpha))$. For example, $\mathsf{br}(13) = 0.1011$. Formally, if $\alpha = \sum_{i=0}^{\infty} 2^i b_i$ with $b_i \in \{0,1\}$, then $\mathsf{br}(\alpha) = \sum_{i=0}^{\infty} b_i/2^{i+1}$.*

*For an integer n, the **Van der Corput set** is the collection of points $p_0, \ldots, p_{n-1}$, where $p_i = (i/n, \mathsf{br}(i))$. See Figure A.1.*

▶ **Lemma 13.** *For a parameter $\varepsilon \in (0,1)$,there is a collection of $O(1/\varepsilon)$ points $P \subset [0,1]^2$ such that any axis-aligned box B with $\mathsf{vol}\big(B \cap [0,1]^2\big) \geq \varepsilon$ contains a point of P.*

Proof. Let $n = \lceil 4/\varepsilon \rceil$. We claim that the Van der Corput set of size n is the desired point set P.

Let B be a box contained in $[0,1]^2$ of width w and height h, with $wh \geq \varepsilon$. Let $q \geq 2$ be the smallest integer such that $1/2^q < h/2 \leq 1/2^{q-1}$. By the choice of q, the projection of B onto the y-axis contains an interval of the form $I = [k/2^q, (k+1)/2^q)$ for some integer k. Let $B_I = B \cap \big\{(x,y) \in [0,1]^2 \;\big|\; y \in I\big\}$ be the box restricted to I along the y-axis. Observe that

$$\mathsf{vol}(B_I) = w/2^q = w/(4 \cdot 2^{q-2}) \geq wh/4 \geq \varepsilon/4 \iff w \geq 2^q\varepsilon/4.$$

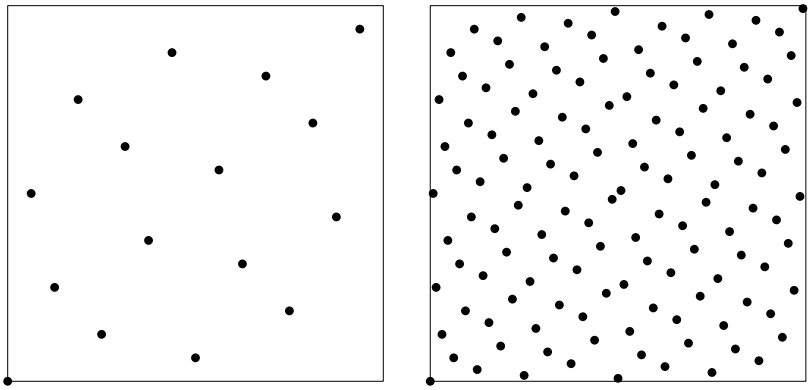


Figure A.1 The Van der Corput set with $n = 16$ (left) and $n = 128$ (right).

Let $S = [0, 1] \times I$, so that each $p_j \in P \cap S$ has $\mathsf{br}(j) \in I$. In particular, the first q binary digits of $\mathsf{br}(j)$ are fixed. This implies that the q least significant binary digits of j are fixed. In other words, $P \cap S$ contains all points p_j such that $j \equiv \ell \pmod{2^q}$ for some integer ℓ – the x-coordinates of the points in P are regularly spaced in the strip S with distance $2^q/n$. If the width of B_I is at least $2^q/n$, then this implies that B contains a point of P in the strip S. Indeed, by the choice of n, $2^q/n \leq 2^q\varepsilon/4 \leq w$. ◀

By extending the definition of the Van der Corput set to higher dimensions, the above proof also generalizes.

▶ **Definition 14** (the Halton-Hammersely set)**.** *For a prime number ρ and an integer $\alpha = \sum_{i=0}^{\infty} \rho^i b_i$, $b_i \in \{0, \ldots, \rho - 1\}$, written in base ρ, define $\mathsf{br}_\rho(\alpha) = \sum_{i=0}^{\infty} b_i/\rho^{i+1}$. Note that $\mathsf{br}_2 = \mathsf{br}$ from Definition 12.*

*For integers n and d, the **Halton-Hammersely set** is the collection of points*

$$p_1, \ldots, p_{n-1},$$

where $p_i = (\mathsf{br}_{\rho_1}(i), \mathsf{br}_{\rho_2}(i), \ldots, \mathsf{br}_{\rho_{d-1}}(i), i/n)$, and $\rho_1, \ldots, \rho_{d-1}$ are the first $d-1$ prime numbers. (Making i/n the dth coordinate instead of the 1st coordinate simplifies future notation.)

▶ **Lemma 15.** *For a parameter $\varepsilon \in (0, 1)$, there is a collection of $2^{O(d \log d)}/\varepsilon$ points $P \subset [0, 1]^d$ such that any axis-aligned box B with $\mathsf{vol}\big(B \cap [0, 1]^d\big) \geq \varepsilon$ contains a point of P.*

Proof. The proof is similar to Lemma 13, with the Chinese remainder theorem as the additional tool.

Let $n = \lceil (2^{d-1}/\varepsilon) \cdot (d-1)\sharp \rceil$, where $k\sharp$ is the **primorial** function, defined as the product of the first k prime numbers. It is known that $k\sharp \leq \exp((1 + o(1))k \log k)$, which implies $n = 2^{O(d \log d)}/\varepsilon$. We claim that the Halton-Hammersely set of size n is the desired point set P.

Denote the side lengths of the box B by $s_1, \ldots, s_d$, with $\prod_{i=1}^{d} s_i \geq \varepsilon$. For each $i = 1, \ldots, d-1$, let q_i be the smallest integer such that $1/\rho_i^{q_i} < s_i/2 \leq 1/\rho_i^{q_i - 1}$, where ρ_i is the ith prime number. By the choice of q_i, the projection of B onto the ith axis contains an interval of the form $I_i = [k_i/\rho_i^{q_i}, (k_i + 1)/\rho_i^{q_i}]$ for some integer k_i. Let S denote the box $I_1 \times \ldots \times I_{d-1} \times [0, 1]$ and $B_S = B \cap S$. Observe that

$$\mathsf{vol}(B_S) = s_d \prod_{i=1}^{d-1} \frac{1}{\rho_i^{q_i}} \geq s_d \prod_{i=1}^{d-1} \frac{s_i}{2\rho_i} \geq \frac{\varepsilon}{2^{d-1}} \prod_{i=1}^{d-1} \frac{1}{\rho_i} \iff s_d \geq \frac{\varepsilon}{2^{d-1}} \prod_{i=1}^{d-1} \rho_i^{q_i - 1}.$$

Similar to Lemma 13, we observe that the point $p_j \in P$ falls into S when $j \equiv \ell_i \pmod{\rho_i^{q_i}}$ for some integers $\ell_1, \ldots, \ell_{d-1}$. By the Chinese remainder theorem, there is exactly one number in the set $\left\{0, 1, \ldots, \prod_{i=1}^{d-1} \rho_i^{q_i} - 1\right\}$ (the dth coordinate of p_j) which satisfies these $d-1$ equations. In particular, the points in $P \cap S$ are spaced regularly along the dth axis with distance $\delta = (1/n) \prod_{i=1}^{d-1} \rho_i^{q_i}$. Once again, we argue that the length of B along the dth axis is at least δ, which implies the result. Indeed, by our choice of n we have that,

$$\delta = \frac{1}{n} \prod_{i=1}^{d-1} \rho_i^{q_i} \leq \frac{\varepsilon}{2^{d-1}} \prod_{i=1}^{d-1} \rho_i^{q_i - 1} \leq s_d. \qquad \blacktriangleleft$$

B Extension: Replacing $[0,1]^d$ with other convex bodies

▶ **Lemma 16.** *Let $\mathcal{C}$ be an arbitrary compact convex body in $\mathbb{R}^d$ with non-empty interior. Suppose there is a (k,ε)-net for volume measure over $[0,1]^d$ of size $T(\varepsilon, k, d)$. For a given integer $k < d$ and $\varepsilon \in (0,1)$, there is a collection of k-flats K of size $T(\Omega_d(\varepsilon), k, d))$, such that any convex body Ξ with $\mathsf{vol}(\Xi \cap \mathcal{C}) \geq \varepsilon \,\mathsf{vol}(\mathcal{C})$ is intersected by a k-flat in K.*

Proof. Assume without loss of generality that $\Xi \subseteq \mathcal{C}$. John's ellipsoid theorem [7] implies that there exists a non-singular affine transformation $\mathbf{M}$, and a ball b of diameter 1, such that $\mathsf{b}/d \subseteq \mathbf{M}(\mathcal{C}) \subseteq \mathsf{b} \subseteq [0,1]^d$, where b/d is b scaled by a factor of $1/d$. We have that $\mathsf{vol}(\mathsf{b}) = c_d 2^{-d}$, where c_d is the volume of the unit ball in $\mathbb{R}^d$. Additionally,

$$\mathsf{vol}\big([0,1]^d\big) = 1 = \frac{2^d}{c_d}\,\mathsf{vol}(\mathsf{b}) = \frac{(2d)^d}{c_d}\,\mathsf{vol}(\mathsf{b}/d) \leq \frac{(2d)^d}{c_d}\,\mathsf{vol}(\mathbf{M}(\mathcal{C})).$$

Set $\delta = c_d/(2d)^d$. Compute a (k,ε')-net for volume measure K over $[0,1]^d$, where $\varepsilon' = \varepsilon/\delta$, which has size $T(\varepsilon', k, d)$. We claim that this is a (k,ε)-net for volume measure with respect to $\mathbf{M}(\Xi)$. Indeed, consider any convex body $\Xi \subseteq \mathcal{C}$ with $\mathsf{vol}(\Xi \cap \mathcal{C}) \geq \varepsilon\,\mathsf{vol}(\mathcal{C})$. Since $\mathbf{M}$ preserves the ratios of volumes, we have that

$$\mathsf{vol}\big(\mathbf{M}(\Xi) \cap [0,1]^d\big) \geq \mathsf{vol}(\mathbf{M}(\Xi) \cap \mathbf{M}(\mathcal{C})) \geq \varepsilon\,\mathsf{vol}(\mathbf{M}(\mathcal{C})) \geq \frac{\varepsilon}{\delta}\,\mathsf{vol}\big([0,1]^d\big) = \varepsilon'\,\mathsf{vol}\big([0,1]^d\big).$$

As such, one of the k-flats in K intersects $\mathbf{M}(\Xi)$. After applying the inverse transformation $\mathbf{M}^{-1}$ to each k-flat in K, one of the k-flats in $\mathbf{M}^{-1}(K)$ intersects Ξ. $\blacktriangleleft$

Reliable Spanners for Metric Spaces

Sariel Har-Peled
Department of Computer Science, University of Illinois, Urbana, IL USA

Manor Mendel
Department of Mathematics and Computer Science, The Open University of Israel, Raanana, Israel

Dániel Oláh
Department of Mathematics and Computing Science, TU Eindhoven, The Netherlands

Abstract

A spanner is reliable if it can withstand large, catastrophic failures in the network. More precisely, any failure of some nodes can only cause a small damage in the remaining graph in terms of the dilation, that is, the spanner property is maintained for almost all nodes in the residual graph. Constructions of reliable spanners of near linear size are known in the low-dimensional Euclidean settings. Here, we present new constructions of reliable spanners for planar graphs, trees and (general) metric spaces.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Spanners, reliability

Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.43

Related Version *Full Version*: https://arxiv.org/abs/2007.08738

Funding *Sariel Har-Peled*: Work on this paper was partially supported by a NSF AF award CCF-1907400.
Manor Mendel: Supported by BSF Grant no. 2018223.
Dániel Oláh: Supported by the Netherlands Organisation for Scientific Research (NWO) through Gravitation-grant NETWORKS-024.002.003.

Acknowledgements The authors thank Kevin Buchin for useful discussions and references.

1 Introduction

Let $\mathcal{M} = (P, \mathrm{d})$ be a finite metric space. Let $G = (P, E)$ be a sparse graph on the points of $\mathcal{M}$ whose edges are weighted with the distances of their endpoints. The graph G is a *t-spanner* (or *t-emulator*) if for any pair of vertices $u, v \in V$ we have $\mathrm{d}_G(u, v) \leqslant t \cdot \mathrm{d}(u, v)$, where $\mathrm{d}_G(u, v)$ is the length of the shortest path between u and v in G, and $\mathrm{d}(u, v)$ is the distance in the metric space between u and v. Spanners were first introduced by Peleg and Schäffer [20] as a tool in distributed computing, but have since found use in other areas of algorithms, networking, data structure and metric geometry, see [19, 18].

Fault tolerant spanners. A desired property of spanners is a resilience to failures of their vertices. The basic notion that captures this is fault tolerance [7, 13, 14, 16, 21]. A graph G is a *k-fault tolerant* t-spanner, if for any subset of vertices B, with $|B| \leqslant k$, the graph $G \setminus B$ is a t-spanner. The disadvantage of k-fault tolerant graphs, is that there is no guarantee if more than k vertices fail, and furthermore, the size of a fault tolerant graph grows (linearly) with the parameter k. In particular, for fixed $t \geqslant 1$, the optimal size of k-fault tolerant $(2t-1)$-spanners on n vertices is $\mathcal{O}(k^{1-1/t} n^{1+1/t})$ [4]. Note that vertex degrees must be at least $\Omega(k)$ to avoid the possibility of isolating a vertex. Thus, it is not suitable for massive failures in the network.

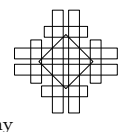


Table 1.1 Our results. Polylog factors are polynomial factors in $\log n$ and $\log \Phi$, where Φ is the spread of the metric. For trees and planar graphs, these results are for graphs with weights on the edges. Here in expectation denotes that the spanner works against an oblivious adversary (here, the expectation is over the randomization in the construction), and the guarantee is on the expected size of the damaged set. Similarly, deterministic implies an adaptive adversary.

Constructions of ϑ-reliable Δ-spanners				
metric	Δ	guarantee	size	ref
Uniform	2	expectation	$\mathcal{O}(n\vartheta^{-1}\log\vartheta^{-1})$	Lemma 4
	t	det. lower bound	$\Omega(n^{1+1/t})$	Lemma 5
	$2t-1$	deterministic	$\mathcal{O}\left(\vartheta^{-2}n^{1+1/t}\right)$	Theorem 9
	$2t$	deterministic	$\mathcal{O}\left(\vartheta^{-1}n^{1+1/t}\right)$	Theorem 9
Finite metrics	$\mathcal{O}(\log n)$	expectation	$\mathcal{O}\left(\vartheta^{-1}n\,\text{polylog}\right)$	Lemma 39
	$\mathcal{O}(t)$	expectation	$\mathcal{O}\left(\vartheta^{-1}n^{1+1/t}\,\text{polylog}\right)$	Lemma 39
	$\mathcal{O}(t\log n)$	deterministic	$\mathcal{O}\left(\vartheta^{-1}n^{1+1/t}\,\text{polylog}\right)$	Lemma 40
	$\mathcal{O}(t^2)$	deterministic	$\mathcal{O}\left(\vartheta^{-1}n^{1+1/t}\,\text{polylog}\right)$	Lemma 40
Ultrametrics	$2+\varepsilon$	expectation	$\mathcal{O}\left(\vartheta^{-1}\varepsilon^{-2}n\,\text{polylog}\right)$	Lemma 41
	$(2+\varepsilon)t-1$	deterministic	$\mathcal{O}\left(\vartheta^{-2}\varepsilon^{-3}n^{1+1/t}\,\text{polylog}\right)$	Lemma 42
Trees	$3+\varepsilon$	expectation	$\mathcal{O}\left(\vartheta^{-1}\varepsilon^{-2}n\,\text{polylog}\right)$	Lemma 43
	$(4+\varepsilon)t-3$	deterministic	$\mathcal{O}\left(\vartheta^{-2}\varepsilon^{-3}n^{1+1/t}\,\text{polylog}\right)$	Lemma 44
Planar graphs	$3+\varepsilon$	expectation	$\mathcal{O}\left(\vartheta^{-1}\varepsilon^{-4}n\,\text{polylog}\right)$	Lemma 45
	$(4+\varepsilon)t-3$	deterministic	$\mathcal{O}(\vartheta^{-2}\varepsilon^{-6}n^{1+1/t}\,\text{polylog})$	Lemma 46

Reliable spanners. An alternative is reliable spanners. For a parameter $\vartheta > 0$, a *ϑ-reliable t-spanner* has the property that for any failure (or attack) set B, the residual graph $G\backslash B$ has a t-spanner path between all pairs of points of $V\backslash B^+$, where $B^+ \supseteq B$ is some set such that $|B^+| \leqslant (1+\vartheta)\,|B|$. We consider two variants. In the standard model (i.e., adaptive adversarial model) the adversary "knows" the spanner G, and the set B is chosen as a worst case for G. In the oblivious (or randomized) case the spanner G is drawn from a probability distribution χ (over the same number of vertices). The adversary knows χ in advance, but not the sampled spanner. In this oblivious model, we require that $\mathbb{E}[|B^+|] \leqslant (1+\vartheta)|B|$. See Section 2 for precise definitions.

Previously, results for reliable spanners were only known in the geometric setting. For any point set $P \subseteq \mathbb{R}^d$, and for any constants $\vartheta, \varepsilon \in (0,1)$, one can construct a ϑ-reliable $(1+\varepsilon)$-spanner with only $\mathcal{O}\big(n\log n\,\text{loglog}^6\, n\big)$ edges [6]. The number of edges can be further reduced by using the relaxed notion of reliable spanners. For any point set $P \subseteq \mathbb{R}^d$ and parameters $\vartheta, \varepsilon \in (0,1)$, one can construct an oblivious $(1+\varepsilon)$-spanner that is ϑ-reliable in expectation and has $\mathcal{O}(n\,\text{loglog}^2\, n\,\text{logloglog}\, n)$ edges [5].

Our results

We provide new constructions of reliable spanners for uniform metrics, finite metrics, ultrametrics, trees and planar graphs. Our new results are summarized in Table 1.1.

Technique. Our approach for constructing reliable spanners is in two steps: We first construct reliable spanners for uniform metrics and then reduce the problem of constructing reliable spanners for general metrics to uniform metrics using **covers**.

Spanners for uniform metrics. Uniform metrics have trivial classical 2-spanners: star graphs. It turns out that in the oblivious model one can simply use "constellation of stars" with a constant number of random centers. I.e. the spanner is linear in size. In the adaptive settings we present a lower bound of $\Omega(n^{1+1/t})$ edges for a reliable t-spanner. We present an asymptotically matching construction of a deterministic $(2t-1)$-reliable spanner with $\mathcal{O}(n^{1+1/t})$ edges. The construction is based on reliable expanders, i.e., expanders that remain expanding under attacks as defined above for spanners. See Section 3.

Covers. A ***t*-cover** of a finite metric space $\mathcal{M} = (P, \mathsf{d})$ is a family of subsets $\mathcal{C} = \{S \mid S \subseteq P\}$, such that for each pair $p, q \in P$ of points there exists a subset in $\mathcal{C}$ that contains both points and whose diameter is at most $t \cdot \mathrm{d}(p,q)$. Covers are used here to extend reliable spanners for uniform metrics into reliable spanners for general metrics. This is done by using spanners for uniform in each set of the cover and then taking a union of the edges of those graphs. See Section 5.

Naturally, the size $\sum_{S\in\mathcal{C}} |S|$ of a t-cover $\mathcal{C}$ is an important parameter in the resulting size of the spanner, so in Section 4 we study good covers. For general n-point spaces with spread at most Φ, we observe that the Ramsey partitions of [17] provide $\mathcal{O}(t)$ covers of size $\mathcal{O}(n^{1+1/t}\log\Phi)$, which is close to optimal, because of an $\Omega(n(n^{1/t}+\log_t\Phi))$ lower bound on the size that we prove. In more specific cases like ultrametrics, trees and planar graphs one can do better. Specifically, for trees and planar graphs one gets $(2+\varepsilon)$-covers of near linear size. For planar graphs, known partitions have much larger gap, which makes these results quite interesting.

New reliable spanners. Plugging the constructions of spanners for uniform metrics with the construction of covers yields reliable spanners for uniform, ultrametric, tree, planar, and general finite metrics. The results are summarized in Table 1.1.

Efficient construction. All our constructions relies on randomized constructions of expanders (over m vertices), that succeeds with probability $\geqslant 1 - 1/m^{O(1)}$. As such, the constructions described can be done efficiently, with potentially an extra $O(\log t)$ factor in the number of edges, if one wants a constructions of spanners, for n vertices, that succeeds with probability $1 - 1/n^{O(1)}$. See Remark 13 for details.

2 Preliminaries

2.1 Metric spaces

For a set $\mathcal{X}$, a function $\mathrm{d} : \mathcal{X}^2 \to [0,\infty)$, is a **metric** if it is symmetric, complies with the triangle inequality, and $\mathrm{d}(p,q) = 0 \iff p = q$. A **metric space** is a pair $\mathcal{M} = (\mathcal{X}, \mathrm{d})$, where d is a metric. For a point $p \in \mathcal{X}$, and a radius r, the **ball** of radius r is the set

$$\mathsf{b}(p,r) = \{q \in \mathcal{X} \mid \mathrm{d}(p,q) \leqslant r\}.$$

For a finite set $X \subseteq \mathcal{X}$, the **diameter** of X is

$$\mathrm{diam}(X) = \mathrm{diam}_{\mathcal{M}}(X) = \max_{p,q\in X} \mathrm{d}(p,q),$$

and the **spread** of X is $\Phi(X) = \frac{\mathrm{diam}(X)}{\min_{p,q\in X, p\neq q}\mathrm{d}(p,q)}$. A metric space $\mathcal{M} = (\mathcal{X}, \mathrm{d})$ is **finite**, if $\mathcal{X}$ is a finite set. In this case, we use $\Phi = \Phi(\mathcal{X})$ to denote the spread of the (finite) metric.

A natural way to define a metric space is to consider an undirected connected graph $G = (P, E)$ with positive weights on the edges. The *shortest path metric* of G, denoted by d_G, assigns for any two points $p, q \in P$ the length of the shortest path between p and q in the graph. Thus, any graph G readily induces the finite metric space $(V(G), \mathrm{d}_G)$. If the graph is unweighted, then all the edges have weight 1.

A **tree metric** is a shortest path metric defined over a graph that is a tree.

2.2 Reliable spanners

▶ **Definition 1.** *For a metric space $\mathcal{M} = (P, \mathrm{d})$, a graph $H = (P, E)$ is a **t-spanner**, if for any $p, q \in P$, we have that $\mathrm{d}(p, q) \leqslant \mathrm{d}_H(p, q) \leqslant t \cdot \mathrm{d}(p, q)$.*

Given a weighted graph $G = (V, E)$, and $B \subseteq V$ we denoted by $G|_B$ the subgraph induced on B. We also use the notation $G \backslash B = G|_{V \backslash B}$.

An **attack** on a graph G is a set of vertices B that fail, and no longer can be used. An attack is **oblivious**, if the set B is picked without any knowledge of G.

▶ **Definition 2** (Reliable spanner)**.** *Let $G = (P, E)$ be a t-spanner for some $t \geqslant 1$ constructed by a (possibly) randomized algorithm. Given an attack B, its **damaged set** B^+ is a set of smallest possible size, such that for any pair of vertices $p, q \in P \backslash B^+$, we have*

$$\mathrm{d}_{G \backslash B}(p, q) \leqslant t \cdot \mathrm{d}(p, q), \tag{2.1}$$

*that is, distances are preserved (up to a factor of t) for all pairs of points not contained in B^+. The quantity $|B^+ \backslash B|$ is the **loss** of G under the attack B. The **loss rate** of G is $\lambda(G, B) = |B^+ \backslash B| / |B|$. For $\vartheta \in (0, 1)$, the graph G is **ϑ-reliable** (in the deterministic or non-oblivious setting), if $\lambda(G, B) \leqslant \vartheta$ holds for any attack $B \subseteq P$. Furthermore, the graph G is **ϑ-reliable in expectation** (or in the oblivious model), if $\mathbb{E}[\lambda(G, B)] \leqslant \vartheta$ holds for any oblivious attack $B \subseteq P$.*

▶ Remark. The damaged set B^+ is not necessarily unique, since there might be freedom in choosing the point to include in B^+ for a pair that does not have a t-path in $G \backslash B$.

Miscellaneous. For a graph G, and a set of vertices $Y \subseteq V(G)$, let

$$\Gamma(Y) = \{x \in V(G) \mid xy \in E(G) \text{ and } y \in Y\}$$

denote the **neighbors** of Y in G.

▶ **Definition 3.** *For a collection of sets $\mathcal{F}$, and an element x, let*

$$\deg(x, \mathcal{F}) = |\{X \in \mathcal{F} \mid x \in X\}|$$

*denote the **degree** of x in $\mathcal{F}$. The maximum degree of any element of $\mathcal{F}$ is the **depth** of $\mathcal{F}$.*

Notations. We use $P + p = P \cup \{p\}$ and $P - p = P \backslash \{p\}$. Similarly, for a graph G, and a vertex p, let $G - p$ denote the graph resulting from removing p.

3 Reliable spanners for uniform metric

Let P be a set of n points and let (P, d) be a metric space equipped with the uniform metric, that is, for all distinct pairs $p, q \in P$, we have that $\mathrm{d}(p, q)$ is the same quantity. Note that $n - 1$ edges are enough to achieve a 2-spanner for the uniform metric by using the star graph.

3.1 A randomized construction for the oblivious case

Construction. Let $\vartheta \in (0,1)$ be a fixed parameter. Set $k = 2\left\lceil \vartheta^{-1} \log \vartheta^{-1} \right\rceil + 1$ and sample k points from P uniformly at random (with replacement). Let $C \subseteq P$ be the resulting set of *center points*. For each point $p \in C$, build the star graph $\ast_p = (P, \{pq \mid q \in P - p\})$, where p is the center of the star. The **constellation** of C is the graph $\ast = \bigcup_{p \in C} \ast_p$, which is the union of the star graphs induced by centers in C.

▶ **Lemma 4** (For the proof, see [11]). *The constellation $\ast$, defined above, is a ϑ-reliable 2-spanner in expectation. The number of its edges is $\mathcal{O}(n\vartheta^{-1} \log \vartheta^{-1})$.*

3.2 Lower bound for a deterministic construction

In the *non-oblivious settings*, the attacker knows the constructed graph G when choosing the attack set B.

▶ **Lemma 5** (For the proof, see [11]). *Let $G = (P, E)$ be a ϑ-reliable t-spanner on P for the uniform metric, where $\vartheta \in (0,1)$ and $t \geqslant 1$. Then, in the non-oblivious settings, G must have $\Omega(n^{1+1/t})$ edges.*

▶ Remark 6. Erdős' girth conjecture states that there exists a graph G with n vertices and $\Omega(n^{1+1/k})$ edges, and girth at least $2k+1$, where the *girth* of G is the length of the shortest cycle in G. The argument in the proof of Lemma 5 is closely related to the standard argument for proving a tight counterpart – any graph with $\Omega(n^{1+1/k})$ edges has girth at most $2k+1$.

3.3 Reliable spanners of the uniform metric for adaptive adversary

Here, we present a construction of reliable spanner that is close to being tight. The spanner is simply a high-degree expander whose properties are described in the following definition.

▶ **Definition 7.** *Denote by $\lambda(G)$ the second eigenvalue of the matrix M/d, where $M = \mathrm{Adj}(G)$ is the adjacency matrix of a d-regular graph G. A **proper expander** specifies a constant $C > 1$, and functions $\mathscr{C}_\delta, \mathscr{c}_\delta > 0$, such that for every $\delta \in (0, 1/4)$ an even integers $d \geqslant \mathscr{C}_\delta$, $n \geqslant d^2$, there exists an n-vertex, d-regular graph $G = (V, E)$, such that:*

(P1) $\forall S \subseteq V,\ |S| \geqslant 12n/(\delta d) \implies |\Gamma_V(S)| > (1-\delta)n$,
(P2) $\forall S \subseteq V,\ |S| \leqslant \mathscr{c}_\delta n/d \implies |\Gamma_V(S)| \geqslant (1-\delta)d|S|$.
(P3) $\lambda(G) \leqslant C/\sqrt{d}$.

For each one of the properties above, it is known that there exists an expander satisfying it: Property (P1) is essentially proved in [6], Property (P2) is folklore, and Property (P3) appears in [8]. Since they hold almost surely for "random regular graphs", they also hold simultaneously. However, we were unable to find in the literature proofs of almost sure existence in the same model of random regular graphs, and contiguity of the different random models does not necessarily hold in the high-degree regime (which is what we need here). Therefore, for completeness, in the full version we reprove (P1) and (P2) in the same random model in which (P3) was proved. We thus get the following in the full version of the paper [11].

▶ **Theorem 8.** *A random graph construction leads to is a proper expander (see Definition 7), asymptotically almost surely.*

With the appropriate choice of parameters, these expanders are reliable spanners for uniform metrics. The proof is somewhat cumbersome and is included in the full version of the paper.

▶ **Theorem 9.** *For every $t \in \mathbb{N}$, $\theta \in (0,1)$, and $n \in 2\mathbb{N}$ such that $n \geqslant e^{\Omega(t)}$, there exist:*

(i) *ϑ-reliable $2t$-spanner with $\mathcal{O}(\vartheta^{-1} n^{1+1/t})$ edges for n-point uniform space, and*

(ii) *ϑ-reliable $(2t-1)$-spanner with $\mathcal{O}(\vartheta^{-2} n^{1+1/t})$ edges for n-point uniform space.*

Applying the above theorem directly on non-uniform metric we obtain the following corollaries.

▶ **Corollary 10.** *Let $\mathcal{M} = (\mathcal{X}, \mathrm{d})$ be a metric space, and let $P \subseteq \mathcal{X}$ be a finite subset of size n. Given parameters $t \in \mathbb{N}$, and $\vartheta \in (0,1)$, there exists a weighted graph G on P, such that:*

(A) *The graph G has $|E(G)| = \mathcal{O}\big(\vartheta^{-2} \cdot n^{1+1/t}\big)$ edges.*

(B) *The graph G is ϑ-reliable. Namely, given any attack set $B \subset \mathcal{X}$, there exists a subset $Q \subseteq P$, such that $|Q| \geqslant |P| - (1+\vartheta)|B \cap P|$. Furthermore, for any two points $p, q \in Q$, we have*

$$\mathrm{d}_{\mathcal{M}}(p,q) \leqslant \mathrm{d}_{G|_Q}(p,q) \leqslant (2t-1) \cdot \mathrm{diam}_{\mathcal{M}}(P),$$

and the path realizing it has at most $(2t-1)$ hops.

In particular, G has hop diameter at most $2t-1$, and diameter at most $(2t-1) \cdot \mathrm{diam}_{\mathcal{M}}(P)$.

▶ **Corollary 11.** *Let $\mathcal{M} = (\mathcal{X}, \mathrm{d})$ be a metric space, and let $P \subseteq \mathcal{X}$ be a finite subset of size n. Given parameters $t \in \mathbb{N}$, and $\vartheta \in (0,1)$, there exists a weighted graph G on P, such that:*

(A) *The graph G has $|E(G)| = \mathcal{O}\big(\vartheta^{-1} \cdot n^{1+1/t}\big)$ edges.*

(B) *The graph G is ϑ-reliable. Namely, given any attack set $B \subset \mathcal{X}$, there exists a subset $Q \subseteq P$, such that $|Q| \geqslant |P| - (1+\vartheta)|B \cap P|$. Furthermore, for any two points $p, q \in Q$, we have*

$$\mathrm{d}_{\mathcal{M}}(p,q) \leqslant \mathrm{d}_{G|_Q}(p,q) \leqslant 2t \cdot \mathrm{diam}_{\mathcal{M}}(P),$$

and the path realizing it has at most $2t$ hops.

In particular, G has hop diameter at most $2t$, and diameter at most $2t \cdot \mathrm{diam}_{\mathcal{M}}(P)$.

3.3.1 A simpler construction of reliable spanner

Here, we prove a somewhat inferior construction of reliable spanner. In particular, the ensuing construction cannot produce reliable expanders with a constant degree but simpler to prove.

The proof will only use Property (P1) of proper expanders from Definition 7. Specifically, it will use the following lemma which is a minor variant of known constructions.

▶ **Lemma 12** ([6])**.** *Let L, R be two disjoint sets, with a total of $n \in 2\mathbb{N}$ elements, and let $\xi \in (0,1)$ be a parameter. There exists a bipartite graph $G = (L \cup R, E)$ with $\mathcal{O}(n/\xi^2)$ edges, such that*

(I) *for any subset $X \subseteq L$, with $|X| \geqslant \xi|L|$, we have that $|\Gamma(X)| > (1-\xi)|R|$, and*

(II) *for any subset $Y \subseteq R$, with $|Y| \geqslant \xi|R|$, we have that $|\Gamma(Y)| > (1-\xi)|L|$.*

▶ Remark 13. The randomized construction of Lemma 12 succeeds with probability $1 - 1/n^{O(1)}$. Since we use the construction below on sets that are polynomially large (i.e., $n^{1/t}$), one can use Lemma 12 constructively in this case (potentially losing an additional $\log t$ factor). This applies to the other expander constructions used in this paper. Note, that while the construction works with high probability, verifying that it indeed works seems computationally intractable.

Construction of the reliable spanner. We define a tree T that contains the n points in its leaves and has exactly $t+1$ levels. Level 0 corresponds to the leaves, and in level k each node has exactly $m = n^{1/t}$ children from level $k-1$, for $k = 1, \ldots, t$. Note that each level corresponds to a partition of P. At level k, each cluster (i.e., tree node) is of size $n^{k/t}$, and there are $n^{1-k/t}$ clusters (i.e., nodes) at this level. For a node $u \in T$, let $v_1, \ldots, v_m$ be the children of u, and let P_u be the set of leaves of the subtree rooted at u. The graph G_u is defined to be the union of bipartite expanders between P_{v_i} and P_{v_j}, for $1 \leqslant i < j \leqslant m$, using Lemma 12 with parameter $\xi = \vartheta/8$. The graph G is the union of graphs G_u for all internal nodes $u \in T$.

Analysis.

▶ **Lemma 14** (For the proof, see [11]). *The graph G, defined above, has $\mathcal{O}(\vartheta^{-2} t \cdot n^{1+1/t})$ edges.*

Fix an arbitrary attack set $B \subseteq P$. A node $u \in T$ is **damaged**, if $|P_u \cap B| \geqslant (1-\vartheta/2)\,|P_u|$ (i.e., most of the nodes in the subtree of u are in the attack set). Let B^+ be all the points of P, that get removed when one removes all the subtrees rooted at damaged nodes.

▶ **Lemma 15** (For the proof, see [11]). *Let B^+ be the set defined above. We have $|B^+| \leqslant (1+\vartheta)\,|B|$.*

Let $\ominus(p,\ell)$ denote the ball of radius ℓ centered at p in the graph $G \backslash B$. For a point $p \in P$, let $u(p,\ell)$ be the node in the tree that is the ancestor of p that is ℓ hops upward toward the root. For a point $p \in P$ and $\ell \geqslant 1$, let

$$\beta(p,\ell) = \left|P_{u(p,\ell)} \cap B\right| \Big/ \left|P_{u(p,\ell)}\right|, \qquad \text{and} \qquad \omega(p,\ell) = \left|P_{u(p,\ell)} \cap \ominus(p,\ell)\right| \Big/ \left|P_{u(p,\ell)}\right|,$$

be the ratio of attack points in $P_{u(p,\ell)}$, and the ratio of ℓ-reachable survivors from p in $P_{u(p,\ell)}$, respectively.

▶ **Lemma 16** (For the proof, see [11]). *For any $p \in P \backslash B^+$, and any $\ell = 1, \ldots, t$, we have $\omega(p,\ell) \geqslant 3\vartheta/8$, where t is the height of T.*

▶ **Proposition 17** (For the proof, see [11]). *For every $t \in \mathbb{N}$, $\theta \in (0,1)$, and $n \in \mathbb{N}$ such that $n^{1/t} \in \{2,3,4,\ldots\}$, there exists ϑ-reliable $(2t-1)$-spanner with $\mathcal{O}\big(\vartheta^{-2} t n^{1+1/t}\big)$ edges for n-point uniform space.*

▶ Remark 18. A constructive version of Proposition 17 requires repeating the construction of the underlying expanders $O(\log t)$ times, so that it succeeds (per such expander) with probability $> 1 - 1/n$. This would yield an extra $O(\log t)$ factor in the number of edges in the graph, see Remark 13.

4 Covers for trees, bounded spread metrics, and planar graphs

▶ **Definition 19.** *For a finite metric space $\mathcal{M} = (P, \mathsf{d})$, a **$t$-cover**, is a family of subsets $\mathcal{C} = \{S_i \subseteq P \mid i = 1, \ldots m\}$, such that for any $p, q \in P$, there exists an index i, such that $p, q \in S_i$, and*

$$\operatorname{diam}(S_i)/t \leqslant \mathsf{d}(p,q) \leqslant \operatorname{diam}(S_i).$$

*The **size** of a cover $\mathcal{C}$ is $\operatorname{size}(\mathcal{C}) = \sum_{S \in \mathcal{C}} |S|$. For a point $p \in P$, its **degree** in $\mathcal{C}$, is the number of sets of P that contain it. The **depth** of $\mathcal{C}$ is the maximum degree of any element of P, and is denoted by $\mathsf{D}(\mathcal{C})$.*

4.1 Lower bounds

Unfortunately, in the worst case, the depth of any cover and its size must depend on the spread of the metric.

▶ **Lemma 20** (For the proof, see [11]). *For any parameter $t > 1$, any integer $h > 1$, and $\Phi = t^h$, there exists a metric $\mathcal{M} = (P, \mathrm{d})$ of n points, such that*

(I) *$\Phi(P) = \Phi$, and*

(II) *any t-cover $\mathcal{C}$ of P has size $\Omega(n \log_t \Phi) = \Omega(nh)$, average degree $\geqslant h/2$, and depth h.*

▶ **Proposition 21** (For the proof, see [11]). *For any $t \in 2, 3, \ldots$, and any sufficiently large n there exists an n-point metric space for which any t-cover must be of size at least $\Omega(n^{1+1/2t})$.*

4.2 Cover for ultrametrics

▶ **Definition 22.** *A* **hierarchically well-separated tree** *(***HST***) is a metric space defined on the leaves of a rooted tree T. To each vertex $u \in T$ there is associated a label $\Delta_u \geqslant 0$. This label is zero for all the leaves of T, and it is a positive number for all the interior nodes. The labels are such that if a vertex u is a child of a vertex v, then $\Delta_u \leqslant \Delta_v$. The distance between two leaves $x, y \in T$ is defined as $\Delta_{\mathrm{lca}(x,y)}$, where $\mathrm{lca}(x, y)$ is the least common ancestor of x and y in T. An HST T is a k-***HST*** if for all vertices $v \in T$, we have that $\Delta_v \leqslant \Delta_{\overline{\mathrm{p}}(v)}/k$, where $\overline{\mathrm{p}}(v)$ is the parent of v in T.*

HSTs are one of the simplest non-trivial metrics, and surprisingly, general metrics can be embedded (randomly) to trees with expected distortion $\mathcal{O}(\log n)$ [2, 9].

▶ **Definition 23.** *A metric $\mathcal{M} = (P, \mathrm{d})$ is an* ***ultrametric****, if for any $x, y, z \in P$, we have that $\mathrm{d}(x, z) \leqslant \max(\mathrm{d}(x, y), \mathrm{d}(y, z))$. Notice, that this is a stronger version of the triangle inequality, which states that $\mathrm{d}(x, z) \leqslant \mathrm{d}(x, y) + \mathrm{d}(y, z)$.*

The following is folklore, easy to verify (see, e.g., [3, Lemma 3.5].

▶ **Lemma 24.** *For $k \geqslant 1$, every finite ultrametric can be k-approximated by a k-HST.*

▶ **Lemma 25** (For the proof, see [11]). *For $k > 1$, every k-HST with spread Φ has 1-cover of depth at most $\log_k \Phi$.*

The following corollary is immediate from the above two lemmas.

▶ **Corollary 26.** *Let $\mathcal{M} = (P, \mathrm{d})$ be an ultrametric over n points with spread Φ. For any $\varepsilon \in (0, 1)$, one can compute a $(1 + \varepsilon)$-cover of $\mathcal{M}$ of depth $\mathcal{O}(\varepsilon^{-1} \log \Phi)$.*

4.3 Cover for general finite metrics

We need the following result.

▶ **Lemma 27** ([17]). *Let (P, d) be an n-point metric space and $k \geqslant 1$. Then there exists a distribution over decreasing sequences of subsets $P = P_0 \supsetneq P_1 \supsetneq \cdots \supsetneq P_s = \varnothing$ (s itself is a random variable), such that for all $\mu > -1/k$, we have $\mathbb{E}\left[\sum_{j=1}^{s-1} |P_j|^{\mu}\right] \leqslant \max\left(\frac{k}{1+\mu k}, 1\right) \cdot n^{\mu+1/k}$, and such that for each $j \in [s]$ there exists an ultrametric ρ_j on P satisfying for every $p, q \in P$, that $\rho_j(p, q) \geqslant \mathrm{d}(p, q)$, and if $p \in P$ and $q \in P_{j-1} \setminus P_j$ then $\rho_j(p, q) \leqslant \mathcal{O}(k) \cdot \mathrm{d}(p, q)$.*

By computing a cover for each ultrametric generated by the above lemma, we get the following.

▶ **Lemma 28** (For the proof, see [11]). *For an n-point metric space $\mathcal{M} = (P, \mathrm{d})$ with spread Φ, and a parameter $k > 1$, one can compute, in polynomial time, a $\mathcal{O}(k)$-cover of $\mathcal{M}$ of size $\mathcal{O}(n^{1+1/k} \log \Phi)$ and depth $\mathcal{O}(kn^{1/k} \log \Phi)$.*

4.4 Covers for trees

Using a tree separator, and applying it recursively, implies the following construction of covers for trees.

▶ **Lemma 29** (For the proof, see [11]). *For a weighted tree metric $T = (P, \mathsf{d})$, with spread Φ, and a parameter $\varepsilon \in (0,1)$, one can compute in polynomial time a $(2+\varepsilon)$-cover of T of depth $\mathcal{O}(\varepsilon^{-1} \log \Phi \log n)$, and size $\mathcal{O}(n\varepsilon^{-1} \log \Phi \log n)$, where $n = |P|$.*

4.5 Covers for planar graphs

Preliminaries. The next lemma can be traced back to the work of Lipton and Tarjan [15], and we include a sketch of the proof for completeness.

▶ **Lemma 30** (For the proof, see [11]). *Let $H = (P, E)$ be a planar triangulated graph with non-negative edge weights. There is a partition of P to three sets X, Y, Z, such that*

- **(i)** *$|X| \leqslant (2/3)n$ and $|Y| \leqslant (2/3)n$,*
- **(ii)** *there is no edge between X and Y, and*
- **(iii)** *Z is composed of two interior disjoint shortest paths that share one of their endpoints, and an edge connecting their other two endpoints.*

▶ **Definition 31.** *For a metric space $(\mathcal{X}, \mathrm{d})$ and a parameter r, an r-net N is a maximal set of points in $\mathcal{X}$, such that*

- **(i)** *for any two net points $p, q \in N$, we have $\mathrm{d}(p,q) > r$, and*
- **(ii)** *for any $p \in \mathcal{X}$, we have $\mathsf{d}(p,N) = \min_{q \in N} \mathrm{d}(p,q) \leqslant r$.*

A net can be computed by repeatedly picking the point furthest away from the current net N, and adding it to the net if this distance is larger than r, and stopping otherwise. We denote a net computed by this algorithm by $\mathrm{net}(\mathcal{X}, r)$.

The following lemma testifies that if we restrict the net to lay along a shortest path in the graph, locally the cover it induces has depth as if the graph was one dimensional.

▶ **Lemma 32** (For the proof, see [11]). *Let G be a weighted graph, and let d be the shortest path metric it induces. Let π be a shortest path in G and let $N = \mathrm{net}(\pi, r) \subseteq \pi$ be a net computed for some distance $r > 0$. For some $R > 0$, consider the set of balls $\mathcal{B} = \{\mathsf{b}(p,R) \mid p \in N\}$. For any point $q \in V(G)$, we have that the degree of q in $\mathcal{B}$ is at most $2R/r + 1$.*

Construction. Let $\varepsilon \in (0,1)$ be an input parameter, and let G be a weighted planar graph. We assume that G is triangulated, as otherwise it can be triangulated (we also assume that we have its planar embedding). Any new edge pq is assigned as weight the distance between its endpoints in the original graph. This can be done in linear time. As usual, we assume that the minimum distance in G is one, and its spread is Φ.

Let Z be the cycle separator given by Lemma 30 made out of two shortest paths π_1 and π_2. Let p_1, p_2, p_3 be the endpoints of these two paths.

For $i = 0, \ldots, m = \lceil \log_{1+\varepsilon/8} \Phi \rceil$, let $N_i = \mathrm{net}(\pi_1, \varepsilon r_i/8) \cup \mathrm{net}(\pi_2, \varepsilon r_i/8) \cup \{p_1, p_2, p_3\}$, where $r_i = (1+\varepsilon/8)^i$. The associated set of balls is

$$\mathcal{B}_i = \{\mathsf{b}(p, (1+\varepsilon/8)r_i) \mid p \in N_i\}.$$

The resulting set of balls is $\mathcal{B}(Z) = \bigcup_i \mathcal{B}_i$. We add the sets of $\mathcal{B}(Z)$ to the cover, and continue recursively on the connected components of $G - Z$. Let $\mathcal{C}$ denote the resulting cover.

Analysis.

▶ **Lemma 33** (For the proof, see [11]). *For any two vertices $p, q \in V(G)$, there exists a cluster $C \in \mathcal{C}$, such that $p, q \in C$, and $\operatorname{diam}(C) \leqslant (2+\varepsilon)\mathrm{d}_G(p,q)$. That is, $\mathcal{C}$ is a $(2+\varepsilon)$-cover of G.*

▶ **Lemma 34** (For the proof, see [11]). *The depth of $\mathcal{C}$ is $\mathcal{O}(\varepsilon^{-2}\log n \log \Phi)$.*

4.5.1 The result

▶ **Theorem 35.** *Let G be a weighted planar graph over n vertices with spread Φ. Then, given a parameter $\varepsilon \in (0,1)$, one can construct a $(2+\varepsilon)$-cover of G with depth $\mathcal{O}(\varepsilon^{-2}\log n \log \Phi)$ in polynomial time.*

▶ Remark 36. It is possible generalize Theorem 35 to the shortest path metric on families of graphs excluding a fixed minor. Specifically, by [12, Lemma 3.3], there exists $O(s^2)$-cover of depth $O(3^s \log \Phi)$ for every metric spaces supported on graph excluding K_s minor and spread Φ. It may be possible to improve the approximation parameter to $O(s)$ using [1], but we have not pursued this avenue. However, this approach can not recover approximation parameter $2+\varepsilon$ for planar graphs as in Theorem 35.

5 From covers to reliable spanners

5.1 The oblivious construction

▶ **Lemma 37** (For the proof, see [11]). *Let $\mathcal{M} = (P, \mathrm{d})$ be a finite metric space, and suppose there exists $\mathcal{C}$, a ξ-cover of $\mathcal{M}$ of size s and depth D. Then, there exists an oblivious ϑ-reliable 2-hop 2ξ-spanner for $\mathcal{M}$, of size $\mathcal{O}\big(s\frac{\mathsf{D}}{\vartheta}\log\frac{\mathsf{D}}{\vartheta}\big)$.*

5.2 The deterministic construction

▶ **Lemma 38** (For the proof, see [11]). *Let $\mathcal{M} = (P, \mathrm{d})$ be a finite metric space over n points, and let $\mathcal{C}$ be a ξ-cover of it of depth D and size s. Then, for any integer $t \geqslant 1$, there exists:*

(A) *A ϑ-reliable $(2t-1)$-hop $(2t-1)\xi$-spanner for $\mathcal{M}$, of size $\mathcal{O}(\vartheta^{-2}\mathsf{D}^2 s n^{1/t})$.*

(B) *A ϑ-reliable $2t$-hop $2t\xi$-spanner for $\mathcal{M}$, of size $\mathcal{O}(\vartheta^{-1}\mathsf{D} s n^{1/t})$.*

5.3 Applications

5.3.1 General metrics

▶ **Lemma 39** (For the proof, see [11]). *Let $\mathcal{M} = (P, \mathrm{d})$ be an n-point metric space of spread at most Φ, and let $\vartheta \in (0,1)$ and $k \in \mathbb{N}$ be parameters. Then, one can build an oblivious ϑ-reliable $\mathcal{O}(k)$-spanner for $\mathcal{M}$ with*

$$\mathcal{O}\Big(\vartheta^{-1} k n^{1+1/k}\log^2\Phi\log\frac{kn^{1/k}\log\Phi}{\vartheta}\Big)$$

edges. In particular, for $k = \log n$, we obtain ϑ-reliable $\mathcal{O}(\log n)$-spanner for $\mathcal{M}$ with

$$\mathcal{O}\big(\vartheta^{-1} n\log n\log^2\Phi(\log\log n + \log\log\Phi + \log(\vartheta^{-1}))\big)$$

edges.

▶ **Lemma 40** (For the proof, see [11]). *Let $\mathcal{M} = (P, \mathrm{d})$ be a finite metric over n points of spread Φ, and let $\vartheta \in (0,1)$ and $k, h \in \mathbb{N}$ int be parameters. Then, one can build a ϑ-reliable $\mathcal{O}(kt)$-spanner for $\mathcal{M}$ with $\mathcal{O}(\vartheta^{-1} k n^{1+1/k+1/t} \log^2 \Phi)$ edges. In particular, when $t = \log n$, we obtain ϑ-reliable $\mathcal{O}(k \log n)$-spanner for $\mathcal{M}$ with $\mathcal{O}(\vartheta^{-1} k n^{1+1/(2k)} \log^2 \Phi)$ edges, and when $t = k$ we obtain ϑ-reliable $\mathcal{O}(t^2)$-spanner for $\mathcal{M}$ with $\mathcal{O}(\vartheta^{-1} t n^{1+1/t} \log^2 \Phi)$ edges.*

5.3.2 Ultrametrics

▶ **Lemma 41** (For the proof, see [11]). *Let $\mathcal{M} = (P, \mathrm{d})$ be an ultrametric over n points with spread Φ, and let $\vartheta, \varepsilon \in (0,1)$ be parameters. Then, one can build an* oblivious *ϑ-reliable $(2+\varepsilon)$-spanner for $\mathcal{M}$ with $\mathcal{O}\big(\vartheta^{-1}\varepsilon^{-2} n \log^2 \Phi \log \frac{\log \Phi}{\vartheta\varepsilon}\big)$ edges.*

▶ **Lemma 42** (For the proof, see [11]). *Let $\mathcal{M} = (P, \mathrm{d})$ be an ultrametric over n points with spread Φ, and let $\vartheta, \varepsilon \in (0,1)$, and $t \in \mathbf{N}$ be parameters. Then, one can build a ϑ-reliable $((2+\varepsilon)t - 1)$-spanner for $\mathcal{M}$ of size $\mathcal{O}(\vartheta^{-2}\varepsilon^{-3} t \cdot n^{1+1/t} \log^3 \Phi)$.*

5.3.3 Tree metrics

▶ **Lemma 43** (For the proof, see [11]). *Let $\mathcal{M} = (P, \mathrm{d})$ be a tree metric over n points with spread Φ, and let $\vartheta, \varepsilon \in (0,1)$ be parameters. Then, one can build an* oblivious *ϑ-reliable $(3+\varepsilon)$-spanner for $\mathcal{M}$ with $\mathcal{O}\big(\vartheta^{-1}\varepsilon^{-2} n \operatorname{polylog}(n, \Phi)\big)$ edges, where $\operatorname{polylog}(n, \Phi) = \log^2 n \log^2 \Phi \log \frac{\log \Phi \log n}{\vartheta\varepsilon}$.*

▶ **Lemma 44** (For the proof, see [11]). *Let $\mathcal{M} = (P, \mathrm{d})$ be a tree metric over n points with spread Φ, and let $\vartheta, \varepsilon \in (0,1)$, and $t \in \mathbf{N}$ be parameters. Then, one can build a ϑ-reliable $((4+\varepsilon)t - 3)$-spanner for $\mathcal{M}$ of size $\mathcal{O}(\vartheta^{-2}\varepsilon^{-3} \cdot n^{1+1/t} \log^3 n \log^3 \Phi)$.*

5.3.4 Planar graphs

▶ **Lemma 45** (For the proof, see [11]). *Let G be a weighted planar graph with n vertices and spread Φ. Furthermore, let $\vartheta, \varepsilon \in (0,1)$ be parameters. Then, one can build an oblivious ϑ-reliable $(3+\varepsilon)$-spanner for G with $\mathcal{O}\big(\vartheta^{-1}\varepsilon^{-4} n \operatorname{polylog}(n, \Phi)\big)$ edges, where $\operatorname{polylog}(n, \Phi) = \log^2 n \log^2 \Phi \log \frac{\log \Phi \log n}{\vartheta\varepsilon}$.*

▶ **Lemma 46** (For the proof, see [11]). *Let G be a weighted planar graph with n vertices and spread Φ. Furthermore, let $\vartheta, \varepsilon \in (0,1)$ and $t \in \mathbf{N}$ be parameters. Then, one can build a deterministic ϑ-reliable $((4+\varepsilon)t - 3)$-spanner for G of size $\mathcal{O}(\vartheta^{-2}\varepsilon^{-6} \cdot n^{1+1/t} \log^3 n \log^3 \Phi)$.*

6 Concluding remarks and open problems

Subsequent work. Recently Filtser and Le [10] improved the results here, getting bounds that do not depend on the spread of the metrics in some cases, for the oblivious adversary case.

Tradeoffs in deterministic constructions for general spaces. Classical spanners are known to have an approximation–size trade-off for general n-point metrics: To achieve $\Theta(t)$ approximation it is sufficient and necessary to have $n^{1+1/t}$ edges in the worst case. In contrast, for reliable spanners we were able only to give an upper bound on the trade-off, with no asymptotically matching lower bound: To achieve $\mathcal{O}(t^2)$ approximation it is sufficient to have $\widetilde{\mathcal{O}}(n^{1+1/t})$ edges. Classically, the uniform metric is 2-approximated by a star graph with only $n-1$ edges. In contrast, we have shown here reliable spanners for uniform metric have approximation–size trade similar to the classical spanner for general metrics. The connection between the two problems is quite intriguing, and is worthy of further research.

The dependence of the size on the spread. The size of spanners constructed in this paper depends on the spread of the metric space. This is because of the reduction to uniform spaces via covers, in which the dependence on the spread is unavoidable in general. However, in some setting this dependence is avoidable. For example [6, 5] achieves this for fixed dimensional Euclidean spaces, and [10] achieves it for doubling spaces, and general finite spaces in the oblivious adversary model. Getting spread-free bounds for the non-oblivious adversary is an interesting problem for further research.

Explicit constructions. To the best of our knowledge, there is no known polynomial time deterministic algorithm for constructing expanders with Property (P1) or Property (P2). Getting such a construction is an interesting open problem.

References

1 Ittai Abraham, Cyril Gavoille, Anupam Gupta, Ofer Neiman, and Kunal Talwar. Cops, robbers, and threatening skeletons: padded decomposition for minor-free graphs. *SIAM J. Comput.*, 48(3):1120–1145, 2019. doi:10.1137/17M1112406.

2 Y. Bartal. On approximating arbitrary metrics by tree metrics. In *Proc. 30th Annu. ACM Sympos. Theory Comput.* (STOC), pages 161–168, 1998. doi:10.1145/276698.276725.

3 Yair Bartal, Nathan Linial, Manor Mendel, and Assaf Naor. On metric Ramsey-type phenomena. *Ann. of Math. (2)*, 162(2):643–709, 2005. doi:10.4007/annals.2005.162.643.

4 Greg Bodwin, Michael Dinitz, Merav Parter, and Virginia Vassilevska Williams. *Optimal Vertex Fault Tolerant Spanners (for fixed stretch)*, pages 1884–1900. Society for Industrial and Applied Mathematics, 2018. doi:10.1137/1.9781611975031.123.

5 K. Buchin, S. Har-Peled, and D. Oláh. Sometimes reliable spanners of almost linear size. *ArXiv*, 2019. (accepted to ESA 2020). arXiv:1912.01547.

6 K. Buchin, S. Har-Peled, and D. Oláh. A spanner for the day after. In *Proceedings of the 35th Symposium on Computational Geometry*, pages 19:1–19:15, 2019. doi:10.4230/LIPIcs.SoCG.2019.19.

7 T.-H. H. Chan, M. Li, L. Ning, and S. Solomon. New doubling spanners: Better and simpler. *SIAM Journal on Computing*, 44(1):37–53, 2015. doi:10.1137/130930984.

8 Ioana Dumitriu, Tobias Johnson, Soumik Pal, and Elliot Paquette. Functional limit theorems for random regular graphs. *Probab. Theory Related Fields*, 156(3-4):921–975, 2013. doi:10.1007/s00440-012-0447-y.

9 J. Fakcharoenphol, S. Rao, and K. Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *Journal of Computer and System Sciences*, 69(3):485–497, 2004. doi:10.1016/j.jcss.2004.04.011.

10 Arnold Filtser and Hung Le. Reliable spanners: Locality-sensitive orderings strike back. *CoRR*, abs/2101.07428, 2021. arXiv:2101.07428.

11 Sariel Har-Peled, Manor Mendel, and Dániel Oláh. Reliable spanners for metric spaces. *CoRR*, abs/2007.08738v2, 2021. arXiv:2007.08738v2.

12 R. Krauthgamer, J. R. Lee, M. Mendel, and A. Naor. Measured descent: A new embedding method for finite metric spaces. *Geom. funct. anal. (GAFA)*, 15(4):839–858, 2005.

13 C. Levcopoulos, G. Narasimhan, and M. Smid. Efficient algorithms for constructing fault-tolerant geometric spanners. In *Proceedings of the 30th ACM Symposium on Theory of Computing*, pages 186–195, 1998. doi:10.1145/276698.276734.

14 C. Levcopoulos, G. Narasimhan, and M. Smid. Improved algorithms for constructing fault-tolerant spanners. *Algorithmica*, 32(1):144–156, 2002. doi:10.1007/s00453-001-0075-x.

15 R. J. Lipton and R. E. Tarjan. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics*, 36(2):177–189, 1979. doi:10.1137/0136016.

16 T. Lukovszki. New results of fault tolerant geometric spanners. In *Proceedings of the 6th Workshop on Algorithms and Data Structures*, pages 193–204, 1999. doi:10.1007/3-540-48447-7_20.

17 M. Mendel and A. Naor. Ramsey partitions and proximity data structures. *J. Euro. Math. Soc.*, 009(2):253–275, 2007. URL: http://eudml.org/doc/277787.

18 G. Narasimhan and M. Smid. *Geometric spanner networks*. Cambridge University Press, New York, NY, USA, 2007.

19 D. Peleg. *Distributed Computing: A Locality-Sensitive Approach*. Society for Industrial and Applied Mathematics, 2000. doi:10.1137/1.9780898719772.

20 D. Peleg and A. Schäffer. Graph spanners. *J. Graph Theory*, 13:99–116, 1989.

21 S. Solomon. From hierarchical partitions to hierarchical covers: Optimal fault-tolerant spanners for doubling metrics. In *Proceedings of the 46th ACM Symposium on Theory of Computing*, STOC '14, page 363–372, 2014. doi:10.1145/2591796.2591864.

A Practical Algorithm with Performance Guarantees for the Art Gallery Problem

Simon B. Hengeveld
Université Rennes 1, France

Tillmann Miltzow
Utrecht University, The Netherlands

Abstract

Given a closed simple polygon P, we say two points p, q see each other if the segment $\text{seg}(p, q)$ is fully contained in P. The art gallery problem seeks a minimum size set $G \subset P$ of guards that sees P completely. The only currently correct algorithm to solve the art gallery problem exactly uses algebraic methods. As the art gallery problem is $\exists\mathbb{R}$-complete, it seems unlikely to avoid algebraic methods, for any exact algorithm, without additional assumptions.

In this paper, we introduce the notion of *vision-stability*. In order to describe vision-stability consider an *enhanced* guard that can see "around the corner" by an angle of δ or a *diminished* guard whose vision is by an angle of δ "blocked" by reflex vertices. A polygon P has vision-stability δ if the optimal number of enhanced guards to guard P is the same as the optimal number of diminished guards to guard P. We will argue that most relevant polygons are vision-stable. We describe a *one-shot vision-stable* algorithm that computes an optimal guard set for vision-stable polygons using polynomial time and solving one integer program. It guarantees to find the optimal solution for every vision-stable polygon. We implemented an *iterative vision-stable* algorithm and show its practical performance is slower, but comparable with other state-of-the-art algorithms. The practical implementation can be found at: `https://github.com/simonheng/AGPIterative`. Our iterative algorithm is inspired and follows closely the one-shot algorithm. It delays several steps and only computes them when deemed necessary. Given a chord c of a polygon, we denote by $n(c)$ the number of vertices visible from c. The *chord-visibility width* ($\text{cw}(P)$) of a polygon is the maximum $n(c)$ over all possible chords c. The set of vision-stable polygons admit an FPT algorithm when parameterized by the chord-visibility width. Furthermore, the one-shot algorithm runs in FPT time when parameterized by the number of reflex vertices.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Art Gallery, Parametrized complexity, Integer Programming, Visibility

Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.44

Related Version *Full Version*: `https://arxiv.org/abs/2007.06920`

Supplementary Material *Software (Source Code)*: `https://github.com/simonheng/AGPIterative` archived at `swh:1:dir:36fab957090b1005b19be68b37849e23e23fe973`

Funding *Simon B. Hengeveld*: MULTIBIOSTRUCT (ANR-19-CE45-0019).
Tillmann Miltzow: NWO Veni grant EAGER.

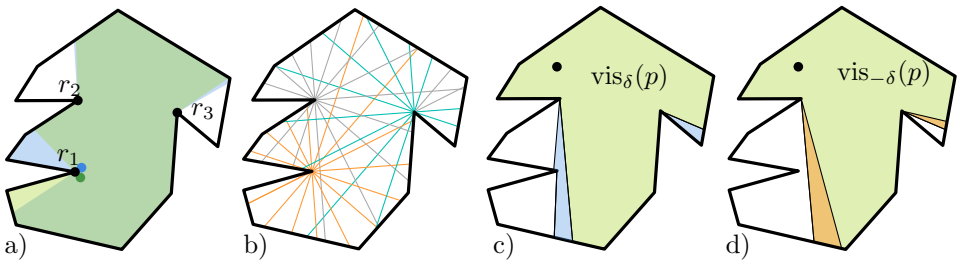


Figure 1 a) A small positional change largely influences how much visibility of the two guards is blocked by r_1, but only has a small effect on the way that r_2 or r_3 blocks the visibility of the guards. b) Shooting rays from reflex vertices. c) Enhanced visibility region. d) Diminished visibility region.

1 Introduction

For most algorithmic problems, there is a *practice-theory gap*. That is a gap between algorithms that perform best in practice and that perform best in theory. A striking example is the euclidean traveling salesperson problem. It has long been known to be NP-hard, see [46] for the most recent hardness results. Despite those theoretical hardness results, in practice, new records of optimally solved large scale instances keep making the news [11].

In this work, we narrow this gap for the art gallery problem. We present several closely related algorithms. Some of them have provable performance guarantees under some mild assumptions. Other algorithms perform comparably well to the best state-of-the-art practical algorithms on the art gallery problem, however at the cost of losing these theoretical performance guarantees.

Motivation. The theoretical study [57] of algorithms has three main motivations:

Prediction: Given a specific algorithm, we want to predict how well it will perform. This can help us to decide if we want to implement it in the first place.

Explanation: Assuming that we know how well an algorithm performs, we want to find an underlying reason for its performance.

Invention: Can we design better algorithms in practice?

These goals are an important measure of success. In an ideal world, theory and practice go hand-in-hand. Theory researchers design and analyze algorithms and practically oriented researchers implement those algorithms and rigorously test them. These tests help to adapt models, assumptions, and performance measures. At the same time, practical researchers may find other approaches fruitful and then in turn theoretical researchers need to find out why those approaches work. This *theory-practice cycle* should ideally lead to a very good theoretical understanding and very fast practical algorithms. Unfortunately, as we will lay out, the study on the art gallery problem has two almost independent lines of research. One solely theoretical and another purely practical one. Thus our theoretical study has limited value to predict, to explain, or to invent. We believe that a solid theoretical understanding will be also useful for experimental research. Our main contribution is to narrow this gap and give a starting point to the theory-practice cycle.

Related work. Let us start by considering practical research. Various researchers implemented algorithms and found the optimal solution on synthetic instances of up to 500 vertices [68, 36, 21, 49, 25, 20, 26, 10, 27, 19]. The idea is to discretize the problem and hope that the solution to the discretized problem also gives the optimal solution to the original problem. To be more specific, the approach is to generate a *candidate* set C and a *witness* set W, then compute the optimal way to guard W using the minimum number of guards in C. In an iterative manner, more candidates and witnesses are generated until the optimal solution is found. While these algorithms *usually* find the optimal solution, there is a simple polygon on which these algorithms run *forever* [2]. We do not know sufficient conditions under which these practical algorithms would give the optimal solution in a finite amount of time.

Let us continue by considering worst-case optimality. We know that the art gallery problem is decidable using tools from real algebraic geometry. The idea is to encode guards by real numbers and use polynomial equations and inequalities to encode visibility [32]. Currently, researchers working on solving polynomial equations repeatedly report that 12 is the maximum number of variables they could handle, using exact methods. (The second author asked this at several conferences and workshops.) To express the art gallery problem by polynomial equations has a considerable blow-up and needs existentially and universally

quantified variables. To summarize, we would be surprised if these exact methods could even find an optimal solution of size two for the art gallery problem. As the art gallery problem is $\exists\mathbb{R}$-complete [3], we know that methods from real algebraic geometry are unavoidable for any exact algorithm, without additional assumptions. The $\exists\mathbb{R}$-completeness of the art gallery problem [3] is a very good explanation of why researchers were not able to find algorithms that avoid the aforementioned algebraic methods. Furthermore, it explains why it is hard to prove that practical discretization schemes work from a theoretical perspective. To be precise, if there would be a discretization scheme with worst-case guarantees then NP $\neq \exists\mathbb{R}$.

To the reader not familiar with the *existential theory of the reals* ($\exists\mathbb{R}$), it may be insightful to get some background information. It is defined in an analogously to NP. Recall that a problem is in NP if for every input there is a binary witness $w \in \{0,1\}^*$ and a verification algorithm that runs on the word RAM in polynomial time. We say a problem is contained in $\exists\mathbb{R}$ if for every input there is a *real* witness $w \in \mathbb{R}^*$ and a verification algorithm that runs on the *real RAM* in polynomial time. Note that the usage of witness in the context of complexity classes is very different from the usage of witness in the context of the art gallery problem [35]. The complexity class $\exists\mathbb{R}$ is important as it gives a precise characterization of many important algorithmic problems [1, 3, 5, 15, 22, 23, 29, 31, 34, 37, 42, 47, 51, 52, 53, 56, 58, 59, 60, 62, 63, 64, 4, 28]. None of these algorithmic problems are known to be contained in NP. The problem to show NP-membership is that it is seemingly impossible to describe a discrete witness of polynomial-size. The complexity-class $\exists\mathbb{R}$ relates the issue for each of those algorithmic problems. To be more specific, either all of those algorithmic problems admit a polynomial size witness or none of them do. This also makes it difficult to discretize any of those problems as any such discretization, usually, would also give rise to a polynomial sized witness.

Maybe the main approach to overcome the discretization problem for the art gallery problem was to consider variants. Specifically, restricting guard placements to the vertices of the polygon avoids the discretization problem, see [55, 14, 45, 7, 54, 38]. While many of these results are very intricate and innovative, we are not aware of any of these algorithms being implemented in practice. We think that there are two main reasons for this. In practice, the discretization problem is solved to a fairly large degree. That is the candidates and witnesses are usually sufficient to determine an optimal solution, even if this cannot be proven, i.e., restricting the guards to the vertices has only a small added benefit. The second reason is that theoretical research focuses on the study of approximation algorithms, which may not be so relevant in practice.

The study of approximation algorithms is mainly motivated by the fact that the art gallery problem and many of its variants are NP-hard [33, 50, 61, 18]. Thus we cannot expect a polynomial-time algorithm, assuming P $\neq$ NP. However, there are four important facts to consider when dealing with the practical performance of approximation algorithms:

Discretization: While the concept of approximation relaxes the worst-case condition, it has not been shown to help sufficiently to find a nice discretization [17]. (See below for a detailed discussion.)

IP-solvers: For some inexplicable reason, once we have a discretization of the art gallery problem, IP-solvers often find the optimum for that discretized problem very fast in practice. (Often only 10% of the total running time is spent on solving IPs [27].) Thus the aim of having a polynomial-time algorithm is not so important.

Visibility: The real bottleneck in practical performance seems to be computing visibilities between candidates and witnesses. Approximation algorithms have to do these computations as well. Thus they may just not be any faster.

Non-optimality: By definition approximation algorithms do not give the optimal solution. This may just be a too high price to pay.

Contribution. In this paper, we introduce the notion of vision-stability. We argue that most practical polygons are vision-stable. Using this assumption, we give a theoretical solution to the discretization problem. Based on this discretization, we develop the *one-shot (vision-stable)* algorithm that is *guaranteed* to find an optimal solution for every vision-stable polygon. The algorithm takes polynomial preprocessing time and thereafter solves one integer program. In particular, this shows that the art gallery problem is in NP for vision-stable polygons. We refine the algorithm and present the *iterative (vision-stable)* algorithm. The iterative algorithm delays many steps of the one-shot algorithm. Both algorithms are *reliable*, in the sense that even if the input polygon is not vision-stable, the reported result is correct. In order to make the iterative algorithm comparable in performance to the state-of-the-art, we cut many corners in our implementation. The downside of this approach is that we are not able to show theoretical performance guarantees for this practical version of the iterative algorithm. On the upside, our implementation has slower but similar performance to previous practical algorithms.

We believe that the concept of vision-stability contributed to all three main goals, as mentioned above. It gives an *explanation* of why the discretization problem is solvable in practice. It led to the *design* of a new practical algorithm, that, as we will see, works quite differently from previous algorithms in many aspects. And finally, it *predicted* correctly that this algorithm is feasible, which we verified in practice.

Let us emphasize here that there is still a large gap, between theory and practice. Our predicted running times are far worse than the observed practical running time. Furthermore, in practice, we used many techniques for which it seems very hard to give a profound theoretical explanation of why they work so well. We hope our work contributes to a deepened theoretical understanding of the art gallery problem.

Vision-stability. Before we formally define the notion of vision-stability, which is the key concept of this paper, let us give some basic intuition on discretizing the art gallery problem. The aim of the discretization process is to define a suitable *candidate set* C such that $\text{opt}[C]$, the optimum guard set restricted to C, is "pretty close" to the actual optimum opt. After defining C in a suitable fashion, we can compute $\text{opt}[C]$ by solving an integer program. We know [17] that the grid $\Gamma = w\mathbb{Z}^2 \cap P$ with a small enough width w is a good such candidate set in the sense that $|\text{opt}[\Gamma]| \leq 10\,|\text{opt}|$, under some mild general position assumptions. Using smoothed analysis [30, 35] and a suitable random model of perturbation, we even know that $|\text{opt}[\Gamma]| = |\text{opt}|$, with high probability. In summary, a fine enough grid contains the optimal solution, under certain assumptions, however, the size of the candidate set C is also important. The grid Γ as described above is huge. Thus computing $\text{opt}[\Gamma]$ is infeasible in practice, making it very desirable to attain a candidate set C with similar properties as Γ and of polynomial size.

As a first step, we realize that a *uniformly distributed* candidate set would either be too large or too coarse. Thus at some spots, we want the candidate set to be denser and at other places in the polygon, we want it to be more sparse. Let us say two guard positions g, g' have almost the same visibility region then hopefully it is not so important to keep track of both positions, but keeping only one of the positions is hopefully sufficient. Thus if the visibility regions of g and g' are very different, we should potentially include both of them in our candidate set C. Another, almost trivial, observation is that a small movement of g towards g' may dramatically change visibility regions, if g and g' are close to a reflex vertex. If g and g' are both very far away from all reflex vertices their visibility regions change almost not at all, see Figure 1 a). Those two observations suggest that we may want to pick a candidate set that is denser closer to reflex vertices and more sparse farther away from reflex vertices.

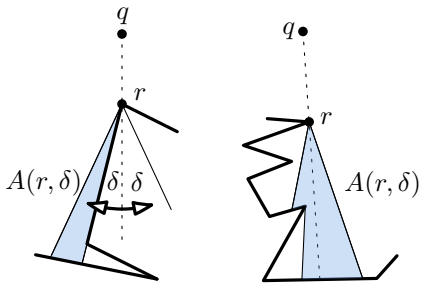


Figure 2 A ray is rotated around a reflex vertex r. It defines a region that is either added or removed from the visibility region.

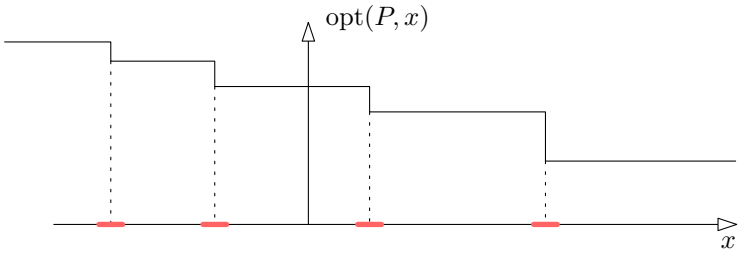


Figure 3 On the x-axis, we have the value by which we either diminish or enhance guards. On the y-axis we display the optimal number of guards. The function $\mathrm{opt}(P, x)$ only takes discrete values and is monotonically decreasing. Thus it has a finite number of breakpoints.

This motivates the following approach. Shoot rays from every reflex vertex, such that the angle between any two rays is at most some given angle δ. This defines an arrangement $\mathcal{A}$. All intersection points of the rays within the polygon P define our candidate set C, see Figure 1 b). On an intuitive level, for every point $p \in P$ there is a candidate $c \in C$ such that the visibility regions of p and c are similar. When r denotes the number of reflex vertices, we get a total number of rays upper bounded by $O(\frac{r}{\delta})$. Thus, the candidate set has size $O(\frac{r^2}{\delta^2})$.

We are now ready for the formal definition of vision-stability. Given a simple polygon P and a point q, the visibility region of q is defined as $\mathrm{vis}(q) = \{x \in P : x \text{ sees } q\}$. Let r be a reflex vertex of P and we assume that q sees r. Then, given some $\delta > 0$, we can define the *visibility enhancing region* $A = A(q, r, \delta)$ as follows. Rotate a ray v with apex r by some angle of δ, clockwise and counter-clockwise. At each time of the rotation, the ray v defines a maximal segment inside P with endpoint r. In some cases, the segment is the single point r, see Figure 2. The region $A(r, \delta)$ is the union of all those segments. For some $\delta > 0$, we define the δ-enhanced visibility region $\mathrm{vis}_\delta(q)$ of q as $\mathrm{vis}(q)$ and for every suitable reflex vertex, we add the region $A(r, \delta)$. We define the δ-diminished visibility region $\mathrm{vis}_{-\delta}(q)$ of q as $\mathrm{vis}(q)$ after we remove the regions $A(r, \delta)$, for every applicable reflex vertex r. To be precise, we define $\mathrm{vis}_\delta(q)$ to be a closed set, both for $\delta > 0$ and $\delta \leq 0$. Given a polygon P, we say that G is *δ-guarding* P if $\bigcup_{g \in G} \mathrm{vis}_\delta(g) = P$. We denote by $\mathrm{opt}(P, \delta)$ the size of the minimum δ-guarding set. For brevity, we denote $\mathrm{opt}(P, 0)$, merely by $\mathrm{opt}(P)$ or, if P is clear from the context, by opt. We say that a polygon P is *vision-stable* or equivalently has vision-stability $\delta > 0$, if $\mathrm{opt}(P, -\delta) = \mathrm{opt}(P, \delta)$. Note that for $\delta' > \delta > 0$, it holds that P has vision-stability δ' implies that P also has vision-stability δ. Thus the smaller the vision-stability the larger we expect the candidate set to be. To avoid confusion, at no time are we interested in actually computing $\mathrm{opt}(P, x)$, for any value $x \neq 0$. The notion of vision-stability is purely a theoretical concept in order to formulate assumptions on the underlying polygon.

Here, we will give a summary of the justification of vision-stability. First, without any assumption, we cannot avoid algebraic methods unless NP = $\exists\mathbb{R}$. Furthermore, there is an argument to be made related to smoothed analysis. Consider $\text{opt}(P, x)$ as a function f of x, see Figure 3. Clearly, f takes only a discrete number of values, by definition. Furthermore, the function is monotonically decreasing, as $\text{vis}_x(p) \subseteq \text{vis}_{x'}(p)$, if $x \leq x'$. Thus the function f has only a finite number of breakpoints. If a breakpoint happens to be at zero then P is not vision-stable. Intuitively, this seems unlikely. See the full version for more details [40].

Discretization. Using vision-stability, we exhibit a candidate set C of polynomial size. The idea is to use the vertices of arrangement $\mathcal{A}$ from Figure 1 b). For technical reasons, we will not use the arrangement $\mathcal{A}$, but a refinement of it. Note that the smaller the vision-stability, the weaker the assumption on the underlying polygon, and thus the larger the required candidate set.

▶ **Theorem 1** (Candidate Set). *Given a polygon with vision-stability δ and r reflex vertices, it is possible to compute a candidate set C (of size $O(\frac{r^4}{\delta^2})$) in polynomial time on a real RAM. The candidate set C contains an optimal solution.*

Although some of the details of the proof are tedious and involved, we see the main contribution on a conceptual level. It is surprising to us that the vision-stability was not formulated earlier. In particular, regarding the popularity of the art gallery problem within computational geometry and the problematic lack of algorithmic results. Let us stress that while we use augmented and diminished visibilities as abstract concepts, we are only interested in solving the original art gallery problem.

Let us give an intuition of the proof idea of Theorem 1. As mentioned before, we use all the vertices of a refinement of the arrangement $\mathcal{A}$, as in Figure 1 b), as our candidate set C. Consider a minimum size set G_0 that is $(-\delta)$-guarding P. Replace each guard $g \in G_0$ by a guard $g' \in C$ that is "close by". This gives a new solution $G_1 \subseteq C$ of equal size. Using that G_0 is $(-\delta)$-guarding, we can show that G_1 is guarding P in the usual sense. As $\text{opt}(P, -\delta) = \text{opt}(P, 0) = \text{opt}(P, \delta)$, we can conclude that $G_1 \subseteq C$ is of minimum size. The technical demanding part of the proof is to show that for every point $p \in P$ there exists a point $c \in C$ such that $\text{vis}_{-\delta}(p) \subseteq \text{vis}(c)$.

Theorem 1 answers an open question posed by several authors of related works [10, 66, 67, 27]. For instance, De Rezende et al. [27] states "*Therefore, it remains an important open question whether there exists a discretization scheme that guarantees that the algorithm always converges [...].*"

In the next paragraph, we discuss how the discretization scheme leads to a correct algorithm that avoids algebraic methods.

One-Shot vision-stable algorithm. Note that in practice there does not exist an algorithm which can be used to compute whether or not a polygon has vision-stability δ. Therefore, it is important that our algorithms work correctly even if the underlying polygon is not vision-stable. Specifically, our algorithms will either compute the optimal solution or report that the input polygon had lower vision-stability than specified by the user. We say our algorithms are *reliable*, as they never return an incorrect answer.

▶ **Theorem 2** (One-Shot vision-stable Algorithm). *Let P be an n vertex polygon, with r reflex vertices. We assume that a suggested value for δ is given as part of the input. Then the one-shot algorithm has a preprocessing time of $O(\frac{r^8}{\delta^4} \log n + n \log n)$ on a real RAM and additionally solves exactly one integer program. The algorithm either returns the optimal solution or reports that the given suggested value for δ is incorrect.*

This is the first algorithm for the classical art gallery problem that avoids using algebraic methods and gives an exact solution.

The core idea of the algorithm is to utilize the candidates from Theorem 1 and use them to build a set-cover instance, which can then be solved using integer programming.

Let us point out that next to vertex-candidates, we also use faces as candidates. This is a distinct feature of our algorithm. All previous algorithms used only points as candidates. In this way, we can easily check that we have not missed a better solution. Say the algorithm returns a solution G of size k and suppose for the purpose of contradiction that there would be a smaller solution G' of size $k' < k$. Pick for each $g \in G'$ a face containing g. (If g lies on an edge or a vertex of $\mathcal{A}$ make an arbitrary choice.) This defines a set $\mathcal{F}$ of faces with $|\mathcal{F}| = k'$. Now we arrive at a contradiction, as $\mathcal{F}$ is a valid guarding of the polygon P. The algorithm primarily minimizes the number of used guards (vertex or face guards), but among the minimum size solutions, it prefers vertex guards over face guards. We say that using vertex guards is a *soft constraint*.

Similar to previous algorithms, we also use witnesses. In the context of the art gallery problem, we require only that all the witnesses are seen, instead of the entire polygon. The hope is that the computed guards will also see the entire polygon. This is a second important step to discretize the problem. In our case, the witness set W will be all *faces* and vertices of some arrangement $\mathcal{A}$. This is a second feature that makes our algorithm unique. As all the faces of $\mathcal{A}$ are covering P, seeing all faces guarantees that P is completely seen. We impose the *hard constraint* that all point-witnesses are seen and the *soft constraint* that each face-witness must be seen by at least one guard.

Due to the hard constraints, we know that our solution is at most the optimal size (theoretically it could be smaller, as face-guards are more powerful than point-guards). If we see all face-witnesses, then we know that the guards are actually guarding P. Furthermore, if all guards are points, we know that we have found a minimum size point guard set. In case that one of the soft-constraints is violated, we will be able to deduce that the underlying polygon was not vision-stable, with the suggested value δ given in the input. The last statement is the technically demanding part of the proof. One of the central concepts of the proof is the *angular capacity of a face*. It measures how small a face is while taking into account the distance to reflex vertices.

For the remainder of the paper, whenever we refer to a candidate, witness or a guard, we could mean either a point or a face, if not further specified.

Parameterized algorithms. As the size of the integer program of the one-shot algorithm only depends on r and $1/\delta$, it exhibits an FPT algorithm, with respect to the number of reflex vertices r, for every fixed δ. The most natural parameter for the art gallery problem is the solution size. As the art gallery problem is W[1]-hard, when parameterized by the solution size [18], research focused on other parameters [9, 7, 8, 12, 43, 44, 6]. Specifically, Agrawal et al. [7] described an elegant FPT algorithm for the art gallery problem. They considered three variants of the art gallery problem defined by restricting guard positions and the part of the polygon that needs to be guarded. By considering the number of reflex vertices as the parameter, they answered a question by Giannopoulos, for those variants. "*Guarding simple polygons has been recently shown to be W[1]-hard w.r.t. the number of (vertex or point) guards. Is the problem FPT w.r.t. the number of reflex vertices of the polygon?*" [39]. We answer the same question, with respect to vision-stable polygons and the classic variant of the art gallery problem.

▶ **Corollary 3** (Reflex-FPT Algorithm)**.** *Given a vision-stable polygon, with any fixed vision-stability. The one-shot algorithm is FPT with respect to the number of reflex vertices.*

Practical algorithm. Although the one-shot algorithm does not require algebraic methods and only polynomial preprocessing time, it is still way too slow to be considered practical. Note that the performance bottleneck is not solving the integer program, but computing visibilities. As a next step, we develop the *iterative vision-stable* algorithm. It is practical and follows the basic principle of the one-shot algorithm. Ideally, we would also like to show provable performance guarantees for the iterative algorithm. Note that the statement that an algorithm is both practical and has theoretical guarantees must be taken with caution. Once we have a *practical* algorithm $\mathcal{P}$ and a *theoretical* algorithm $\mathcal{T}$, we can easily get a third algorithm $\mathcal{B}$ that has *both* properties as follows. Run algorithm $\mathcal{P}$ within the running time bound of $\mathcal{T}$. If $\mathcal{P}$ does not return a solution abort and run $\mathcal{T}$. Here, $\mathcal{T}$ serves as a *safe guard* for $\mathcal{P}$. Clearly $\mathcal{B}$ performs as well in practice as $\mathcal{P}$ and has the theoretical bounds of $\mathcal{T}$. Thus, when we say that we show theoretical performance guarantees of some algorithm, we should really ask ourselves, if we show those guarantees for the algorithm that we actually use or for some safe guards that aren't ever used in practice.

The core idea of the iterative algorithm is as follows. We start with a very coarse arrangement $\mathcal{A}$. Using the faces and the vertices of $\mathcal{A}$, we define a candidate set C and a witness set W. We compute which candidates see which witnesses. This enables us to build an integer program, that tries to find a minimum guard set $G \subseteq C$, as described for the one-hot algorithm. Note that vertices and faces may serve as guards and some face-witnesses may be unguarded. As a secondary objective function, the integer program tries to minimize the number of face-guards used in G and the number of unseen face-witnesses. If the guard set G contains only point guards and sees all face-witnesses, the iterative algorithm reports the optimal solution. Otherwise, it refines the arrangement $\mathcal{A}$ and goes to the next iteration.

Irrational-Guard Polygon. In Figure 4 we show the first 8 iterations of the Irrational-Guard polygon [2]. The orange points and faces represent point- and face-guards in the intermediate solution. The green faces represent faces not fully seen by the current candidate solution. Both orange and green faces are split in the next iteration. Note that for each of the orange and green faces, we draw a random vertex in the same colour, to make also very small faces visible.

Local complexity. One of the bottlenecks is the large number of possible visibilities between candidates and witnesses that we have to compute. Due to the low local complexity of the input polygons, most of those pairs are not seeing each other. We exploit this by building a so-called *weak visibility polygon tree*. In this tree, any point p in node $n(p)$ can see a point q in node $n(q)$, if the two nodes are siblings or in a parent-child relationship, see Figure 5.

Before, we can describe the weak visibility polygon tree, recall that the weak visibility polygon $\text{vis}(s)$ (of a segment s) is defined by $\text{vis}(s) = \{p \in P : \exists q \in s \text{ s.t. } \text{seg}(p,q) \subseteq P\}$. To build the weak visibility polygon tree T of P, we start with an arbitrary edge e on the boundary of P. We compute the weak visibility polygon $\text{vis}(e)$ of e, which is the root of T. For every edge e' of $\text{vis}(e)$, which is not part of the boundary of P, we compute the weak visibility polygon $\text{vis}(e')$ with respect to the polygon $P \setminus \text{vis}(e)$. Those weak visibility polygons are the children of $\text{vis}(e)$. We continue recursively to compute the children of every weak visibility polygon. Note that every node of T is a weak visibility polygon W of some defining chord c.

Figure 4 The first 8 iterations of the Irrational-Guard polygon [2].

Interestingly, this inspired a new structural parameter, which we call the chord-visibility width. Given a chord c of P, we denote by $n(c)$ the number of vertices visible from c. The *chord-visibility width* ($\mathrm{cw}(P)$) of a polygon is the maximum $n(c)$ over all possible chords c.

We show that the art gallery problem is FPT with respect to the chord-visibility width.

▶ **Theorem 4** (Chord-Width-FPT). *Let P be a simple polygon with vision-stability at least some fixed δ. Then there is an FPT algorithm for the art gallery problem with respect to the chord-visibility width.*

The core idea is to use dynamic programming along the weak visibility polygon tree, similar to dynamic programming algorithms for tree-width. The challenge here is to find bounds on the number of candidates per node.

Critical witnesses. Another practical idea is to reduce the total number of witnesses that we use. Instead of using all faces and vertices of $\mathcal{A}$ as witnesses, we only use some random selection, which we call *critical witnesses*. We use heuristics to update this critical witness set. In deciding whether to add a critical witness or not, we are faced with some trade-offs. If we add very few critical witnesses, we have to make more loops until the IP solution returns a guard set that sees the entire polygon. If we add too many critical witnesses, the set of critical witnesses grows unnecessarily fast and we have to compute many more visibilities.

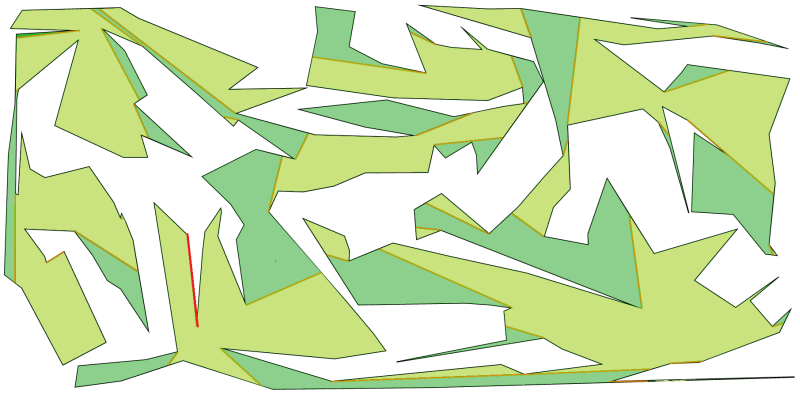


Figure 5 The polygon together with a weak visibility polygon tree. The polygon has 200 vertices, but each node in the weak visibility polygon tree has only about 20 vertices. The red segment indicates the starting edge of the weak visibility polygon tree.

Visibility queries. One of the major bottlenecks at the beginning of this project was the computation of weak visibility queries. That is, we often need to decide if a given face sees a given point or another face. In a follow-up project [41], we developed a fast practical algorithm to compute weak visibility polygons.

Losing performance guarantees. The main reason that the iterative algorithm does not have the same performance guarantees as the one-shot algorithm is as follows. It is possible that the iterative algorithm keeps splitting a certain face (and its children) many times, only to conclude much later that it was misled. It could have found a solution much earlier by splitting one of the larger faces. In practice, it seems usually a very good idea to split the faces that were selected by the Integer program solution. Especially, if those faces are small, we made progress and avoided usually unnecessary splits of big unimportant faces. It is easily possible to design an algorithm in a way that it will not split too small faces, before also splitting occasionally bigger faces, that might be useful. In this way, we are still able to ensure theoretical performance guarantees, see Theorem 5. However, this theorem adds little to the goals of explanation, prediction, and invention. Quite the opposite, this analysis suggests that faces should be split in a way that is harmful to practical performance. In the full version [40], we describe several different versions of the iterative algorithm. When we use the safeguard version of the algorithm, we get the following theorem.

▶ **Theorem 5** (Iterative Algorithm). *Let P be an n vertex polygon, with vision-stability δ. Then the iterative algorithm returns the optimal solution to the art gallery problem. It has a running time of $(\frac{n}{\delta})^{O(1)} + T$ per iteration and takes at most $(\frac{n}{\delta})^{O(1)}$ iterations. Here T denotes the time it takes to solve one integer program.*

Experimental results. We implemented and tested the iterative algorithm with a 64-bit Windows 10 operating system, an 8-core Intel(R) Core i7-7700HQ CPU at 2800 Mhz and 16 GB of main memory. The practical implementation makes heavy use of version 4.13.1 of CGAL [65]. The IP solver used was IBM ILOG CPLEX version 12.10 [16].

We compared our implementation directly with the algorithm from Tozoni et al. [68], as this is the currently best algorithm, for which there was freely accessible code available. In order to get a deeper understanding, we analyzed various aspects of the running time. One

of them is the distribution of the running time w.r.t. different subroutines. Furthermore, we studied the influence of vision-stability on the running time. We also study the effect of our speed-up methods. Lastly, we study the iterative algorithm on the irrational-guard polygon [2]. For an in-depth description of our experiments with all details, we refer the reader to the full version [40]. Here, we only highlight the most important findings.

Comparison. We tested our algorithm and the algorithm of Tozoni et.al. on 5 sets of 30 polygons of sizes 60, 100, 200, and 500 vertices. The results can be seen in Table 1. We see that, except for size 60, our algorithm is slightly faster. However, we want to point out that comparing those running times should be taken with a grain of salt. Both algorithms rely on a software environment that is not identical. To some degree, the differences in the running time may come from performance differences from this software. First note, that Tozoni et al. implemented their algorithm in Linux, whereas, we implemented our algorithm in Windows. Interestingly, CGAL runs between factor $2-3$ faster on Linux compared to Windows [48]. Secondly, our algorithm has to compute visibilities of faces instead of point visibilities. Thirdly, while testing the implementation of Tozoni et al, we could not use the best available IP solver, which might skew the results a bit. Overall, our algorithm seems faster for larger polygons, but does not make a very large improvement.

Table 1 A comparison of the iterative algorithm without safe guards with the results from Tozoni et al. [68], both the results reported by Tozoni et al. themselves [68] and results found using their implementation on our hardware.

Sizes	Average time (s)		
	Tozoni et al.	**Tozoni et al. (Our hardware)**	**The Iterative Algorithm**
60	0.26	0.18	0.39
100	0.94	0.68	0.52
200	3.77	2.54	2.02
500	35.04	22.34	18.2

CPU distribution. We analyzed the distribution of the CPU time of the iterative algorithm. The results are shown below in a pie-chart. We see that solving integer programs is the dominating factor of the running time. This shows that to improve the running time of the algorithm, we must reduce the total number or the size of the IPs. Alternatively, we can optimize the IP solver, or perhaps experiment with different IP solvers.

It may appear strange that we spend so much energy on reducing the CPU time spent on speeding up visibility queries when the CPU usage is dominated by solving IPs. The reason is that before we implemented all of the improvements described before, the running time was dominated by weak-visibility queries.

In general, the CPU distribution needs to be regarded with a grain of salt. It seems that there is usually one subroutine that dominates the running time. Often conceptual improvements decrease the running time by several factors, which in turn makes a different subroutine appear to dominate the running time. Thus the fact that solving IPs says more about the components that we optimized rather than which parts are inherently more difficult. Thus we consider it a success that the running time is now dominated by solving IPs.

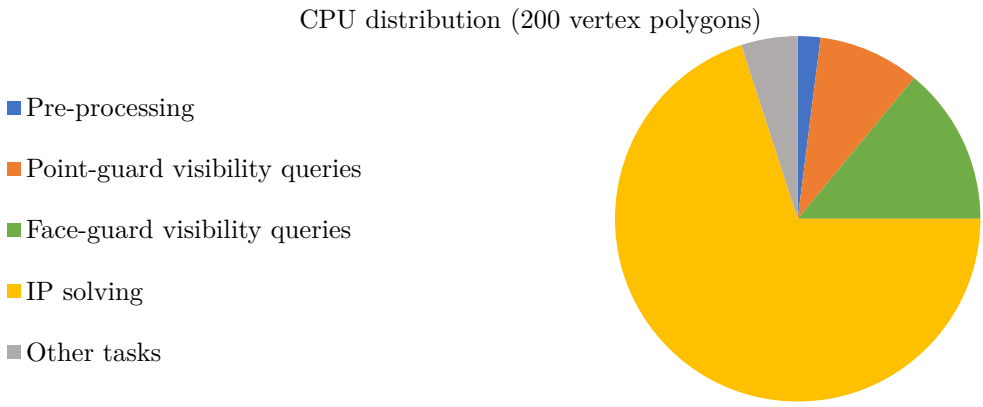


Figure 6 The chart shows the CPU distribution of the Iterative Algorithm implementation for solving 30 polygons of size 200.

Vision-stability. Unfortunately, we cannot measure the vision-stability of a polygon nor approximate it. To get a vague idea of the influence of the vision-stability on the running time, we define the granularity of the subdivision at the end of the iterative algorithm. The granularity is related to the smallest face in the final subdivision. Interestingly, even with polygons of the same size the running times vary widely. The large correlations between the observed running times and the granularity indicate that vision-stability may have a significant influence on the total running time.

Table 2 The correlation coefficients between the measured minimum granularity and the running time, computed per size.

Size	60	100	200	500
Correlation	0.07	0.32	0.21	0.76

Weak visibility polygon tree. To verify the amount of visibility queries that we save by using the weak visibility polygon tree, we conducted an experiment. Note that computing the weak visibility polygon tree requires the computation of weak visibility polygon. In practice, this was achieved by using an efficient new algorithm, about which a follow-up paper will be published [41]. The precise percentage highly depends on the type of polygon. See Table 3 for a detailed overview. Interestingly, the number of reflex vertices per node of the weak visibility polygon grows much slower than the input size. This indicates that chord-visibility width may be a useful practical parameter to study for geometric algorithms in polygons.

Table 3 We tested 30 input polygons from the AGPLIB library [24] of four sizes. For each size class we see the averages of characteristics of the weak visibility polygon trees.

Size	60	100	200	500
Tree size	14.2	23.0	46.3	115.0
Largest polygon	20.5	23.3	26.2	28.4
Largest number of reflex vertices	5.9	6.2	7.0	9.2
Percentage of queries saved	16.7%	35.4%	63.5%	87.3%

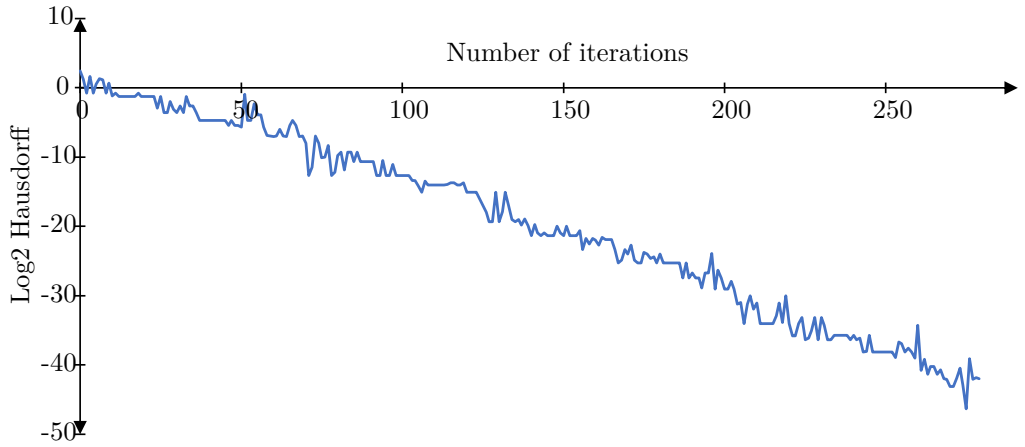


Figure 7 The iterative algorithm based on the notion of vision-stability reports a sequence of solutions. The Graph shows on the x-axis the iterations from 1 to about 300 and on the y-axis, the $\log_2$ of the Hausdorff distance to the optimal solution.

Irrational guards. We show that the implementation of our algorithm provides a rapidly improving solution even for polygons that are not vision-stable. Specifically, Abrahamsen et al. [2] introduced a small and simple polygon, which requires irrational guards for an optimal guarding using point-guards. Although, the iterative algorithm avoids irrational numbers it still returns a guard set G_i for each iteration $i = 1, 2, 3, \ldots$. Recall that G_i consists of faces and points. As we know the optimal solution G^*, we can compute $d_i = d(G^*, G_i)$, see Figure 7.

Future research. The vast majority of the work on the art gallery problem focused on variants of the classic question. There are almost no positive theoretical algorithmic results on the original art gallery problem, with some exceptions [13, 32, 17]. We believe that the main reason for this focus on variants is the fact that the art gallery problem is inherently continuous, as is reflected by its $\exists\mathbb{R}$-completeness [3]. Now that we arguably broke that barrier, we hope that more progress will be made on the original problem.

- Can we adapt the algorithm to polygonal domains with holes? Here, the main bottleneck seems to be adapting the visibility queries to polygons with holes.
- Does the iterative algorithm always converge towards the optimal solution, even if the underlying polygon is not vision-stable? Our experimental results suggest that we converge exponentially fast to an optimal solution. It is intriguing to see if this also holds true in general.
- One simple way to make the one-shot algorithm more practical would be to find a smaller witness and candidate set. What is the smallest integer program that guarantees to give the optimal solution for vision-stable polygons? The bound we gave seems to have plenty of room for improvement.

References

1 Zachary Abel, Erik D. Demaine, Martin L. Demaine, Sarah Eisenstat, Jayson Lynch, and Tao B. Schardl. Who needs crossings? Hardness of plane graph rigidity. In *32nd International Symposium on Computational Geometry (SoCG 2016)*, pages 3:1–3:15, 2016.

2 Mikkel Abrahamsen, Anna Adamaszek, and Tillmann Miltzow. Irrational guards are sometimes needed. In *SoCG 2017*, pages 3:1–3:15, 2017. `arXiv:1701.05475`.

3 Mikkel Abrahamsen, Anna Adamaszek, and Tillmann Miltzow. The art gallery problem is $\exists\mathbb{R}$-complete. In *STOC 2018*, pages 65–73, 2018. arXiv:1704.06969, doi:10.1145/3188745.3188868.

4 Mikkel Abrahamsen, Linda Kleist, and Tillmann Miltzow. Training neural networks is er-complete. *arXiv*, 2021. arXiv:2102.09798.

5 Mikkel Abrahamsen, Tillmann Miltzow, and Nadja Seiferth. A framework for $\exists\mathbb{R}$-completeness of two-dimensional packing problems. *FOCS*, 2020.

6 Akanksha Agrawal, Pradeesha Ashok, Meghana M. Reddy, Saket Saurabh, and Dolly Yadav. FPT algorithms for conflict-free coloring of graphs and chromatic terrain guarding. *Arxiv*, 1905.01822, 2019. arXiv:1905.01822.

7 Akanksha Agrawal, Kristine V. K. Knudsen, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. The Parameterized Complexity of Guarding Almost Convex Polygons. In *SoCG 2020*, LIPIcs, pages 3:1–3:16, 2020. doi:10.4230/LIPIcs.SoCG.2020.3.

8 Akanksha Agrawal, Sudeshna Kolay, and Meirav Zehavi. Parameter analysis for guarding terrains. In *SWAT*, 2020.

9 Akanksha Agrawal and Meirav Zehavi. Parameterized analysis of art gallery and terrain guarding. In *International Computer Science Symposium in Russia*, pages 16–29. Springer, 2020.

10 Yoav Amit, Joseph S.B. Mitchell, and Eli Packer. Locating guards for visibility coverage of polygons. *International Journal of Computational Geometry & Applications*, 20(05):601–630, 2010.

11 David Applegate, Robert Bixby, Vašek Chvátal, and William Cook. *TSP in practice*, 2021. URL: http://www.math.uwaterloo.ca/tsp/index.html.

12 Pradeesha Ashok and Meghana Reddy. Efficient guarding of polygons and terrains. In *International Workshop on Frontiers in Algorithmics*, pages 26–37. Springer, 2019.

13 Patrice Belleville. Computing two-covers of simple polygons. Master's thesis, McGill University, 1991.

14 Pritam Bhattacharya, Subir Kumar Ghosh, and Sudebkumar Prasant Pal. Constant approximation algorithms for guarding simple polygons using vertex guards. *arXiv*, 2017. arXiv:1712.05492.

15 Daniel Bienstock. Some provably hard crossing number problems. *Discrete & Computational Geometry*, 6(3):443–459, 1991.

16 Robert Bixby. IBM CPLEX. URL: https://www.ibm.com/analytics/cplex-optimizer.

17 Édouard Bonnet and Tillmann Miltzow. An approximation algorithm for the art gallery problem. In *SoCG 2017*, pages 20:1–20:15, 2017. arXiv:1607.05527, doi:10.4230/LIPIcs.SoCG.2017.20.

18 Édouard Bonnet and Tillmann Miltzow. Parameterized hardness of art gallery problems. *ACM Transactions on Algorithms*, 16(4), 2020. doi:10.1145/3398684.

19 Dorit Borrmann, Pedro J. de Rezende, Cid C. de Souza, Sándor P. Fekete, Stephan Friedrichs, Alexander Kröller, Andreas Nüchter, Christiane Schmidt, and Davi C. Tozoni. Point guards and point clouds: solving general art gallery problems. In *SoCG*, pages 347–348. ACM, 2013. doi:10.1145/2462356.2462361.

20 Andrea Bottino and Aldo Laurentini. A nearly optimal sensor placement algorithm for boundary coverage. *Pattern Recognition*, 41(11):3343–3355, 2008.

21 Andrea Bottino and Aldo Laurentini. A nearly optimal algorithm for covering the interior of an art gallery. *Pattern Recognition*, 44(5):1048–1056, 2011.

22 Jean Cardinal, Stefan Felsner, Tillmann Miltzow, Casey Tompkins, and Birgit Vogtenhuber. Intersection graphs of rays and grounded segments. *Journal of Graph Algorithms and Applications*, 22:273–295, 2018.

23 Jean Cardinal and Udo Hoffmann. Recognition and complexity of point visibility graphs. *Discrete & Computational Geometry*, 57(1):164–178, 2017.

24 Marcelo C. Couto, Pedro J. de Rezende, and Cid C. de Souza. Instances for the Art Gallery Problem, 2009. URL: www.ic.unicamp.br/~cid/Problem-instances/Art-Gallery.

25 Marcelo C. Couto, Pedro J. de Rezende, and Cid C. de Souza. An exact algorithm for minimizing vertex guards on art galleries. *International Transactions in Operational Research*, 18(4):425–448, 2011.

26 Marcelo C. Couto, Cid C. de Souza, and Pedro J. de Rezende. Experimental evaluation of an exact algorithm for the orthogonal art gallery problem. In *International Workshop on Experimental and Efficient Algorithms*, pages 101–113. Springer, 2008.

27 Pedro J. de Rezende, Cid C. de Souza, Stephan Friedrichs, Michael Hemmer, Alexander Kröller, and Davi C. Tozoni. Engineering art galleries. *Algorithm Engineering*, pages 379–417, 2016. doi:10.1007/978-3-319-49487-6_12.

28 Argyrios Deligkas, John Fearnley, and Themistoklis Melissourgos. Square-cut pizza sharing is ppa-complete. *arXiv preprint*, 2020. arXiv:2012.14236.

29 Michael G. Dobbins, Linda Kleist, Tillmann Miltzow, and Paweł Rzążewski. $\forall\exists\mathbb{R}$-completeness and area-universality. *WG 2018*, 2018. arXiv:1712.05142.

30 Michael Gene Dobbins, Andreas Holmsen, and Tillmann Miltzow. Smoothed analysis of the art gallery problem. *arXiv*, 2018. arXiv:1811.01177.

31 Michael Gene Dobbins, Andreas Holmsen, and Tillmann Miltzow. A universality theorem for nested polytopes. *arXiv*, 2019. arXiv:1908.02213.

32 Alon Efrat and Sariel Har-Peled. Guarding galleries and terrains. *Inf. Process. Lett.*, 100(6):238–245, 2006. doi:10.1016/j.ipl.2006.05.014.

33 Stephan Eidenbenz, Christoph Stamm, and Peter Widmayer. Inapproximability results for guarding polygons and terrains. *Algorithmica*, 31(1):79–113, 2001.

34 Jeff Erickson. Optimal curve straightening is $\exists\mathbb{R}$-complete. *arXiv*, 2019. arXiv:1908.09400.

35 Jeff Erickson, Ivor van der Hoog, and Tillmann Miltzow. Smoothing the gap between NP and $\exists\mathbb{R}$. *accepted to FOCS 2020*, 2020. arXiv:1912.02278.

36 S. Friedrichs. Integer solutions for the art gallery problem using linear programming. Masterthesis, 2012.

37 Jugal Garg, Ruta Mehta, Vijay V. Vazirani, and Sadra Yazdanbod. ETR-completeness for decision versions of multi-player (symmetric) Nash equilibria. In *ICALP 2015*, pages 554–566, 2015.

38 Subir Kumar Ghosh. Approximation algorithms for art gallery problems in polygons. *Discrete Applied Mathematics*, 158(6):718–722, 2010.

39 Panos Giannopoulos. Open problems: guarding problems, 2016.

40 Simon Hengeveld and Tillmann Miltzow. A practical algorithm with performance guarantees for the art˜ gallery problem. *arXiv*, 2020. arXiv:2007.06920.

41 Simon Hengeveld, Tillmann Miltzow, and Frank Staals. Weak visibility by convex expansion. in preparation 2020.

42 Ross J. Kang and Tobias Müller. Sphere and dot product representations of graphs. In *SoCG*, pages 308–314. ACM, 2011.

43 Farnoosh Khodakarami, Farzad Didehvar, and Ali Mohades. A fixed-parameter algorithm for guarding 1.5 d terrains. *Theoretical Computer Science*, 595:130–142, 2015.

44 Farnoosh Khodakarami, Farzad Didehvar, and Ali Mohades. 1.5 d terrain guarding problem parameterized by guard range. *Theoretical Computer Science*, 661:65–69, 2017.

45 David G. Kirkpatrick. An $O(\lg\lg \mathrm{OPT})$-approximation algorithm for multi-guarding galleries. *Discrete & Computational Geometry*, 53(2):327–343, 2015. doi:10.1007/s00454-014-9656-8.

46 Sándor Kisfaludi-Bak, Jesper Nederlof, and Karol Węgrzycki. A gap-eth-tight approximation scheme for euclidean tsp. *arXiv preprint*, 2020. arXiv:2011.03778.

47 Linda Kleist. *Planar graphs and faces areas – Area-Universality*. PhD thesis, Technische Universität Berlin, 2018. PhD thesis.

48 Bernhard Kornberger. Why is cgal significantly slower under windows? URL: https://stackoverflow.com/questions/58008543/.

49 Alexander Kröller, Tobias Baumgartner, Sándor P. Fekete, and Christiane Schmidt. Exact solutions and bounds for general art gallery problems. *Journal of Experimental Algorithmics (JEA)*, 17:2–3, 2012.

50 Der-Tsai Lee and Arthur K. Lin. Computational complexity of art gallery problems. *IEEE Transactions on Information Theory*, 32(2):276–282, 1986. doi:10.1109/TIT.1986.1057165.

51 Anna Lubiw, Tillmann Miltzow, and Debajyoti Mondal. The complexity of drawing a graph in a polygonal region. *Arxiv*, 2018. Graph Drawing 2018.

52 Colin McDiarmid and Tobias Müller. Integer realizations of disk and segment graphs. *Journal of Combinatorial Theory, Series B*, 103(1):114–143, 2013.

53 Nicolai E Mnëv. The universality theorems on the classification problem of configuration varieties and convex polytopes varieties. In Oleg Y. Viro, editor, *Topology and geometry – Rohlin seminar*, pages 527–543. Springer-Verlag Berlin Heidelberg, 1988.

54 Rajeev Motwani, Arvind Raghunathan, and Huzur Saran. Covering orthogonal polygons with star polygons: The perfect graph approach. *J. Comput. Syst. Sci.*, 40(1):19–48, 1990. doi:10.1016/0022-0000(90)90017-F.

55 Joseph O'Rourke. *Art Gallery Theorems and Algorithms*. Oxford University Press, 1987.

56 Jürgen Richter-Gebert and Günter M. Ziegler. Realization spaces of 4-polytopes are universal. *Bulletin of the American Mathematical Society*, 32(4):403–412, 1995.

57 Tim Roughgarden. Beyond the worst-case analysis of algorithms (introduction). *CoRR*, abs/2007.13241, 2020. arXiv:2007.13241.

58 Marcus Schaefer. Complexity of some geometric and topological problems. In *Proceedings of the 17th International Symposium on Graph Drawing (GD 2009)*, volume 5849 of *Lecture Notes in Computer Science (LNCS)*, pages 334–344. Springer, 2009.

59 Marcus Schaefer. Realizability of graphs and linkages. In *Thirty Essays on Geometric Graph Theory*, pages 461–482. Springer, 2013.

60 Marcus Schaefer and Daniel Štefankovič. Fixed points, Nash equilibria, and the existential theory of the reals. *Theory of Computing Systems*, 60(2):172–193, 2017. doi:10.1007/s00224-015-9662-0.

61 Dietmar Schuchardt and Hans-Dietrich Hecker. Two NP-hard art-gallery problems for ortho-polygons. *Math. Log. Q.*, 41:261–267, 1995. doi:10.1002/malq.19950410212.

62 Yaroslav Shitov. A universality theorem for nonnegative matrix factorizations. *arXiv*, 2016. arXiv:1606.09068.

63 Yaroslav Shitov. The complexity of positive semidefinite matrix factorization. *SIAM Journal on Optimization*, 27(3):1898–1909, 2017.

64 Peter Shor. Stretchability of pseudolines is np-hard. *Applied Geometry and Discrete Mathematics-The Victor Klee Festschrift*, 1991.

65 The CGAL Project. *CGAL User and Reference Manual*. CGAL Editorial Board, 4.1.3 edition, 2020. URL: https://doc.cgal.org/4.1.3/Manual/packages.html.

66 Davi C. Tozoni, Pedro J. de Rezende, and Cid C de Souza. A practical iterative algorithm for the art gallery problem using integer linear programming. *Optimization Online*, 2013.

67 Davi C. Tozoni, Pedro J. de Rezende, and Cid C. de Souza. The quest for optimal solutions for the art gallery problem: A practical iterative algorithm. In *Experimental Algorithms*, pages 320–336, 2013.

68 Davi C. Tozoni, Pedro J. de Rezende, and Cid C. de Souza. Algorithm 966: A practical iterative algorithm for the art gallery problem using integer linear programming. *ACM Trans. Math. Softw.*, 43(2), 2016. doi:10.1145/2890491.

Approximate Range Counting Under Differential Privacy

Ziyue Huang
Hong Kong University of Science and Technology, Hong Kong, China

Ke Yi
Hong Kong University of Science and Technology, Hong Kong, China

Abstract

Range counting under differential privacy has been studied extensively. Unfortunately, lower bounds based on discrepancy theory suggest that large errors have to be introduced in order to preserve privacy: Essentially for any range space (except axis-parallel rectangles), the error has to be polynomial. In this paper, we show that by allowing a standard notion of geometric approximation where points near the boundary of the range may or may not be counted, the error can be reduced to logarithmic. Furthermore, our approximate range counting data structure can be used to solve the approximate nearest neighbor (ANN) problem and k-NN classification, leading to the first differentially private algorithms for these two problems with provable guarantees on the utility.

2012 ACM Subject Classification Theory of computation → Computational geometry; Security and privacy → Formal methods and theory of security

Keywords and phrases Differential Privacy, Approximate Range Counting

Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.45

Related Version *Full Version*: `http://www.cse.ust.hk/~yike/DPRangeCount.pdf`

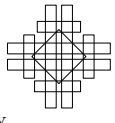
Funding This work has been supported by HKRGC under grants 16202317, 16201318, and 16201819.

1 Introduction

Differential privacy (DP) [7] is a rigorous notion of privacy-preserving data publishing, which has been widely adopted. Essentially, it ensures that no individual can substantially change the probability distribution on the published results of an analysis. A central problem studied under differential privacy is *counting queries*. Given a set of points $P \subseteq \mathcal{U}$, where $\mathcal{U}$ is the universe, the goal is to release a differentially private data structure of P, so that for any query $Q \subseteq \mathcal{U}$ from a certain query family $\mathcal{Q}$, we can find the count[1] $|P \cap Q|$. To ensure the privacy of P, some noise has to be injected to the query answers, so the key challenge is to minimize the error under a given privacy budget (ε, δ); please see Section 2.1 for a more formal DP definition and the meaning of the privacy parameters ε, δ.

When $\mathcal{Q}$ consists of arbitrary subsets of $\mathcal{U}$, then the optimal error achievable is $\sqrt{n} \cdot \mathrm{poly}(\frac{1}{\varepsilon}, \log \frac{|\mathcal{U}|\cdot|\mathcal{Q}|}{\delta})$ [12]. In the other extreme when $\mathcal{Q} = \mathcal{U}$, which are called *point queries*, also known as the *histogram problem*, the error is $O(\frac{1}{\varepsilon} \cdot \min\{\log|\mathcal{U}|, \log\frac{1}{\delta}\})$ [22]. This is a large gap. Thus, there have been a lot of interest in studying specific query families, and particularly, identifying those that yield polylogarithmic errors. In this paper, we study query families in constant-dimensional Euclidean space. More precisely, the dataset P consists of n points from $\mathcal{U} = [u]^d$, where $[u]$ denotes integers from 1 to u, while $\mathcal{Q}$ consists of geometric ranges of a certain type, e.g., (axis-parallel) rectangles, halfspaces, simplices, spheres, or arbitrarily-shaped regions. In the geometric setting, $\mathcal{Q}$ is often called a *range space*.

[1] We allow P to be a multiset, i.e., the same point from $\mathcal{U}$ may appear multiple times in P. In this case, $|P \cap Q|$ adds up all the multiplicities of points of P that are in Q.



1.1 Previous work

For $d = 1$, the only interesting range space is the set of intervals. For this case, Dwork et al. [8] present an $(\varepsilon, 0)$-DP algorithm with error $O\left(\frac{1}{\varepsilon}\log^{1.5} u\right)$, as well as a lower bound of $\Omega(\frac{1}{\varepsilon}\log u)$[2]. Bun et al. [4] show that under (ε, δ)-DP, this lower bound can be broken, achieving error $2^{(1+o(1))\log^* u} \cdot \log(1/\delta)/\varepsilon$. They also show an $\Omega(\log^* u \cdot \log(1/\delta)/\varepsilon)$ lower bound for $\exp(-\varepsilon n/\log^* n) \le \delta \le 1/n^2$. Note that these lower bounds justify the restriction that the points must have integer coordinates. In addition, they hold irrespective of the space/time of the data structure. They solely rely on the privacy requirement.

For $d \ge 2$, Chan et al. [5] extend the algorithm of [8] to answer axis-parallel rectangle queries with error $O\left(\frac{1}{\varepsilon}\log^{1.5d} u\right)$, which is later improved to $O\left(\frac{1}{\varepsilon}(\log u + \log^{1.5d} n)\right)$ [9]. These results show that rectangles are "easy", namely, they admit polylogarithmic errors. Unfortunately, it turns out they are essentially the only easy cases. Muthukrishnan and Nikolov [18] and Nikolov et al. [19] build an equivalence between the error and the *discrepancy* of the range space $\mathcal{Q}$. Rectangle range counting has polylogarithmic errors essentially because the discrepancy of rectangles is polylogarithmic [17]. On the other hand, since the discrepancy of other natural range spaces, such as halfspaces, simplices, and spheres, is all $n^{\phi(d)}$, where $\phi(d) \in [\frac{1}{4}, \frac{1}{2}]$ is a constant depending on the particular range space and d [17], they are ruled out for having polylogarithmic errors. Making things worse, if we allow nonconvex ranges, the discrepancy becomes $\sqrt{n}$, which means that the aforementioned solution that works for arbitrary query families [12] is already optimal, namely, geometry doesn't help.

However, a series of differentially private range counting data structures [6, 14, 23, 16, 20, 21, 24] have been proposed by practitioners, mostly based on hierarchical space decompositions. They work well on many real-world datasets, but perform badly on high-discrepancy point sets, as predicted by the lower bounds [18, 19].

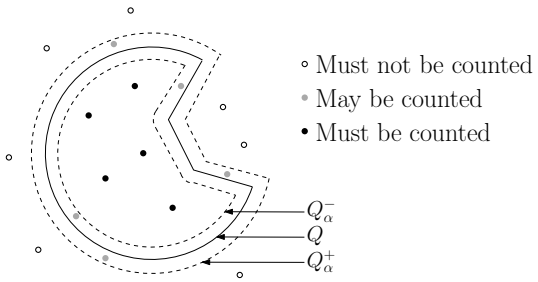


Figure 1 Approximate range counting.

1.2 Our results

This paper provides the theoretical justification that geometry *does* help, at least in constant dimensions. However, in order to circumvent the polynomial discrepancy lower bounds, we have to introduce some relaxation. Specifically, we consider *approximate range counting* as defined in [1]. The *diameter* of a range Q is the largest distance between any two points in Q. Given a range Q of diameter w and a constant *fuzziness parameter* $0 < \alpha < 1$, the inner range Q_α^- is defined as the region of points whose distance from any point exterior to Q is at

[2] All upper bounds stated in this paper hold for any single query with constant probability, while lower bounds hold for the maximum error of all queries with constant probability.

least αw, while the outer range Q_α^+ is those points whose distance from a point interior to Q is at most αw. Then any count between $|P \cap Q_\alpha^-|$ and $|P \cap Q_\alpha^+|$ is considered a valid answer to Q, i.e., points in $Q_\alpha^+ - Q_\alpha^-$ may or may not be counted (see Figure 1). In fact, Blum et al. [3] used a similar notion of geometric approximation with differential privacy, but their error is still polynomial.

We show that polylogarithmic errors are achievable under this notion of approximation. In particular, we show that (1) under $(\varepsilon, 0)$-DP, the error is $O\left(\frac{1}{\varepsilon}\log u\right)$, with a lower bound of $\Omega\left(\frac{1}{\varepsilon}\log u\right)$; and (2) under (ε, δ)-DP, the error can be reduced to $O(\frac{1}{\varepsilon}(\log\log u + \log\frac{1}{\delta} + \log^{1.5} n))$, with a lower bound of $\Omega\left(\frac{1}{\varepsilon}\log^* u \cdot \log(1/\delta)\right)$ for $\exp(-\varepsilon n/\log^* n) \le \delta \le 1/n^2$. These results hold for *any* range space in constant dimensions (even if the ranges are nonconvex). Compared with the $\sqrt{n}$ lower bound mentioned above, we obtain this exponential improvement exactly due to geometry – the notion of fuzziness has no counterpart in arbitrary query families over an abstract set system. Technically, the fuzziness allows a packing argument (see Theorem 5), which we borrow from the non-private setting [1].

In practice, this notion of fuzziness is often acceptable, considering that either data or the range (or both) are often imprecise themselves. For example, if the public want to know how many people got COVID-19 in their neighborhood, whether the cases near the boundary are included or not is not very important. Another way of comparing our result with prior work on DP range counting (which does not allow fuzziness) is that we have an extra $|P \cap (Q_\alpha^+ - Q_\alpha^-)|$ term in our error bound. This term is incomparable to $\sqrt{n}$, so our result doesn't contradict the discrepancy-based lower bound [18, 19]. However, this error term is small on most real-world datasets, which explains why small errors are often observed in many practical solutions.

Our data structures are naturally based on approximate range counting structures in the non-private setting, in particular, the BBD-tree [1]. However, some non-trivial modifications and analyses are needed to make the BBD-tree private, especially for reducing the dependence on u in the (ε, δ)-DP case.

Our approximate range counting data structure can also be used to solve the approximate nearest neighbor (ANN) problem and k-NN classification, yielding the first DP algorithms for these two problems with provable guarantees on the utility. For the ANN problem, returning the NN itself (or any other point in P) is not private; instead, we return a distance that approximates the nearest distance. Returning only the distance can still be useful in many applications, e.g., if the user only wants to know whether there is a point in P that is sufficiently close to her query point. Also, only the distance to the query point is needed in k-NN classification; please see Section 4 for details.

2 Preliminaries

2.1 Differential privacy

Let $P \sim P'$ denote two neighboring databases, i.e., one contains one more point than the other.

▶ **Definition 1** (Differential Privacy [10]). *For $\varepsilon > 0$ and $\delta \ge 0$, a randomized algorithm $\mathcal{M}$ is (ε, δ)-differentially private if for any neighboring databases $P \sim P'$ and any $S \subseteq \mathsf{Range}(\mathcal{M})$,*

$$\Pr[\mathcal{M}(P) \in S] \le e^\varepsilon \cdot \Pr[\mathcal{M}(P') \in S] + \delta$$

The case when $\delta = 0$ is also referred to as *pure differential privacy*. In practice, ε is usually a constant ranging from 0.1 to 10, while δ must be much smaller than $1/n$. For a numeric query f, the most common technique for designing DP mechanisms is by masking the result

with Laplace noise calibrated to the *sensitivity* of the query $\Delta_f = \max_{P \sim P'} |f(P) - f(P')|$. We use $\mathrm{Lap}(\lambda)$ to denote the Laplace distribution with parameter λ, which has probability density function $\Pr[X = z] = \frac{1}{2\lambda} \exp(-|z|/\lambda)$ and variance $2\lambda^2$. And the fact that $\Pr[|X| > t \cdot \lambda] \leq \exp(-t)$ is often useful to analyze the accuracy of the Laplace mechanism.

▶ **Theorem 2** (Laplace Mechanism [10]). *For a numeric query f over a database P, an $(\varepsilon, 0)$-differentially private mechanism is to output $f(P) + X$, where $X \sim Lap(\Delta_f/\varepsilon)$.*

We need the following properties of DP mechanisms.

▶ **Theorem 3** (Basic Composition [10]). *Let $\mathcal{M}_i$ be an (ε, δ)-differentially private mechanism for all $i \in [k]$, then $\mathcal{M}(P) = (\mathcal{M}_1(P), \mathcal{M}_2(P), \ldots, \mathcal{M}_k(P))$ is $(k\varepsilon, k\delta)$-differentially private.*

▶ **Theorem 4** (Group Privacy [10]). *Let $\mathcal{M}$ be an $(\varepsilon, 0)$-differentially private mechanism. For any two databases P and P' that differ by at most k individuals and any $S \subseteq \mathsf{Range}(\mathcal{M})$,*

$$\Pr[\mathcal{M}(P) \in S] \leq e^{k\varepsilon} \cdot \Pr[\mathcal{M}(P') \in S]$$

2.2 The BBD-tree

The *balanced box-decomposition tree (BBD-tree)* [2] is a hierarchical space decomposition[3] where each node is associated with a region of space called a *cell*. We define a *box* to be an axis-parallel rectangle whose aspect ratio is either 1 or 2. Recall that points in P have integer coordinates. To avoid the ambiguity of points lying on the boundary of a box, the boxes' corners all have coordinates in the form $x + \frac{1}{2}$ for integer x. Each cell in the BBD-tree is either a box or the region between two boxes, one enclosed within the other. Thus each cell comprises of an outer box and an optional inner box. For a cell c, its size is the length of the longest side of its outer box. We use $\mathrm{size}(c)$ to denote its size and $\mathrm{count}(c)$ to denote the number of points in P lying inside c. The aspect ratio of each cell in the BBD-tree is bounded by 2, which allows us to bound the number of cells that intersect any range of a given diameter.

▶ **Theorem 5** ([1]). *Consider any space decomposition of height h, consisting of cells of bounded aspect ratios. Let C be any subset of its cells, where each cell has size at least s and they all intersect a range of diameter w. Then $|C| = O(h + (w/s)^d)$. If the cells are also disjoint, then $|C| = O((w/s)^d)$.*

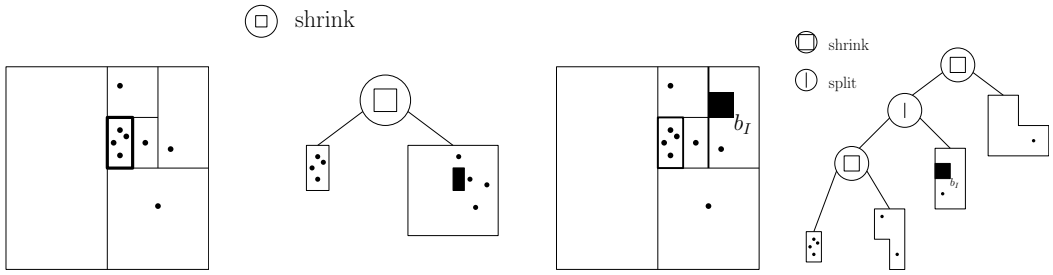
Note that the quadtree has this property (where the aspect ratios are all 1), but it has a large height $h = O(\log u)$. On the other hand, the BBD-tree has only $O(\log n)$ height. To achieve this without violating the aspect ratio constraint, the key idea is to give a little more flexibility to the shape of the cells.

The BBD-tree is a binary tree and is constructed by the repeated application of two partitioning operations alternatively on each level, *split* and *shrink*. Starting from the root node whose corresponding cell is the entire space $[u]^d$, we recursively divide each cell c by either split or shrink, until $\mathrm{size}(c)$ or $\mathrm{count}(c)$ is at most 1.

To split a cell, we bisect it along its longest side (it is guaranteed that the bisecting hyperplane does not intersect c's inner box if it has one). The resulting cells have aspect ratio of either 1 or 2. Now consider the shrink operation. If c does not have an inner box,

[3] As a hierarchical space decomposition naturally corresponds to a tree, where each node corresponds to a cell, we will use the terms "node" and "cell" interchangeably.

the shrink operation is performed by repeatedly applying split operations and recursing on the child box b_c with the majority of points in c, until $\text{count}(b_c \cap c) \leq \frac{2}{3}\text{count}(c)$. Then all intermediate splits are discarded, and the two children of c are b_c and $c - b_c$ (i.e., c and b_c are the outer and inner boxes of the cell $c - b_c$; see Figure 2a). For the case when there is an inner box b_I in c, we first follow the same procedure to obtain the box b_c. If $b_I \subseteq b_c$, then we shrink c to $b_c - b_I$ (the other child is thus $c - b_c$). Otherwise, suppose the majority boxes obtained during the series of splits are $b_1, \ldots, b_k = b_c$. Let b_j be the smallest box in the sequence that contains b_I. Then we first shrink c to $b_j - b_I$ (the other child is $c - b_j$), then split $b_j - b_I$ to b_{j+1} and $b_j - b_{j+1} - b_I$, then shrink b_{j+1} to b_c (the other child is $b_{j+1} - b_c$), as shown in Figure 2b. The shrink operations ensure that $\text{count}(c)$ decreases by a constant factor as we descend a constant number of levels, leading to an $O(\log n)$ height of the BBD-tree.



(a) shrink without an existing inner box **(b)** shrink with an existing inner box b_I

Figure 2 The shrink operation in the BBD-tree.

3 Approximate range counting

In this section, we describe and analyze our private range counting data structure. It is similar to the existing heuristic solutions [6, 20, 24], namely, we also release a hierarchical space decomposition that is differentially private and can be used to answer any range query. Each cell c in the space decomposition is associated with a count masked by Laplace noise, which we denote by noisy_count(c). However, there are two differences, which are important for achieving theoretical guarantees: First, we use a different query procedure, which is described in Section 3.1. Second, we use the BBD-tree as our space decomposition, and we show how to make it private in Sections 3.2 and 3.3.

3.1 Query procedure

Our query procedure is similar to the one in [1]. Given a space decomposition and a range Q, we visit its cells in a top-down manner, summing up the noisy counts of cells as we go along. We stop exploring further at a cell in the following two cases: (1) if $c \cap Q_\alpha^- = \emptyset$, we just skip c; and if (2) if $c \subseteq Q_\alpha^+$, we add noisy_count(c) and then skip c. If we do not skip a cell, we visit its children recursively. The detailed query procedure is given in Algorithm 1.

It is clear that line 1 in Algorithm 1 does not introduce any error. Line 2 introduces a zero-mean error with magnitude proportional to the noise level, while line 3 introduces a bias equal to $\text{count}(c)$. We will account for these two sources of errors when analyzing the accuracy of our private BBD-tree.

Algorithm 1 Approximate Range Counting Procedure: Query(Q, α, c).

Input: a range Q; range approximation factor α; a cell c in the space decomposition.
Output: an answer to the α-approximate range query Q.

```
1: if c ∩ Q_α^- = ∅ then return 0
2: if c ⊆ Q_α^+ then return noisy_count(c)
3: if c is a leaf cell then return 0
4: s ← 0
5: for each child c' of c do
6:     s ← s + Query(Q, α, c')
7: return s
```

3.2 Pure differential privacy

For pure DP, we use the BBD-tree with only split operations, which results in a full (binary) quadtree. The quadtree has u^d leaves and $O(\log u)$ height. It is safe to release this tree structure as it does not depend on P. It is clear that the sensitivity of count(c) is 1 for each cell c and each point contributes to the counts of $d \log u$ nodes on a root-to-leaf path. Then by the basic composition theorem, adding noise drawn from $\mathrm{Lap}(d \log u/\varepsilon)$ to each count(c) is sufficient to preserve $(\varepsilon, 0)$-DP.

Now we analyze the error of an approximate range counting query. The proofs of the following theorem, as well as some others, are given in the full version [15].

▶ **Theorem 6.** *There is an $(\varepsilon, 0)$-differentially private space decomposition, such that any α-approximate range counting query can be answered within error $O\left(\varepsilon^{-1}\alpha^{-d/2} \log u \log \frac{1}{\beta}\right)$ with probability at least $1-\beta$. Moreover, it can be pruned to $O(n)$ nodes with probability at least $1-\beta$, and in this case the error is $O\left(\varepsilon^{-1}\alpha^{-d} \log u \log \frac{n}{\beta}\right)$.*

The $\log u$ dependency of the error follows from the height of the tree. One may wonder if we could make the shrink operation differentially private, thus reducing its height to $O(\log n)$. Unfortunately, we show that this is not possible under pure DP, by presenting an $\Omega(\log u)$ lower bound on the error for any pure DP approximate range counting algorithm (see Section 5).

3.3 (ε, δ)-differential privacy

By adopting (ε, δ)-DP, we can improve the error dependence on u from $\log u$ to $\log \log u$. The key, as mentioned above, is to make the shrink operations private. Recall that the original shrink algorithm on a cell c uses a series of splits until the majority box contains at most $\frac{2}{3}$count(c) points. To make this comparison private and ensure that the shrink algorithm still decreases count(c) by a constant factor, we can add noise proportional to count(c). However, directly using count(c) to calibrate the noise would not be differentially private. In fact, even the total count, i.e., n, cannot be released; so we use $\tilde{n} := n + \frac{4}{\varepsilon}\log\frac{2}{\beta} + \mathrm{Lap}(\frac{4}{\varepsilon})$ instead. The idea is to make $O(\log \tilde{n})$ "guesses" for the right noise level, which is captured by a parameter h. At $h = 0$, the magnitude of noise is $\tilde{n}^{O(1)}/\varepsilon$, and it will exponentially decrease as h increases; at $h = O(\log \tilde{n})$, the noise magnitude will become $O(1/\varepsilon)$.

We now describe how to construct a private BBD-tree. We still use shrink and split operations on alternating levels of the tree. The split operation is the same as in the standard BBD-tree, while the private shrink operation is shown in Algorithm 2 for a given cell c with noise level h. The shrink operation on the root node is invoked with $h = 0$; on a non-root

node c, we will use $h = h_p + 1$, where h_p is the noise level used in the shrink operation at the grandparent of c. We stop further subdividing c when $\text{size}(c) = 1$, when we are about to invoke the shrink algorithm with $h = H := \lceil 2.5 \log \tilde{n} \rceil$, or after the shrink algorithm returns `LEAF`.

Algorithm 2 works as follows. If $\text{count}(c)$ is too small compared with the the current noise level (determined probabilistically), the algorithm will return `AGAIN`. In this case, we invoke the algorithm again with $h \leftarrow h + 1$, decreasing the noise magnitude by a constant factor. Otherwise it will split c repeatedly and recurse on the majority box. Compared with the non-private shrink operation, we make the following changes. Let b_c be the current majority box. First, we add noise before checking if $\text{count}(b_c \cap c) \leq \frac{2}{3}\text{count}(c)$. As a result, we cannot guarantee that $\text{count}(c)$ will decrease by a constant factor, but it will be the case with high probability. Second, in the non-private case, we can just return b_c when $\text{count}(b_c \cap c) \leq \frac{2}{3}\text{count}(c)$ after the last split. We can no longer do this, as being the majority box is also sensitive information which depends on P (this was not sensitive before the splits stop, as the count of the majority box was large enough to hide the presence or absence of one data point). Instead, we return b_c or its sibling after a noisy comparison. Finally, when c has an inner box b_I, the non-private shrink algorithm will never try to split b_I because the count would have become $0 \leq \frac{2}{3}\text{count}(c)$ already. However, as the comparison with $\frac{2}{3}\text{count}(c)$ is now probabilistic, we need to guard against this from happening. In particular, when the algorithm tries to split b_I, it will return `LEAF`. If the algorithm does not return `AGAIN` or `LEAF`, it will return a box b_c. Then we shrink c to b_c as in the standard BBD-tree as described in Section 2.2; this may involve an interleaving split in case c has an inner box b_I that is disjoint from b_c.

Algorithm 2 Private Shrink Operation; $\text{Shrink}(c, h, \beta)$.

Input: an input cell $c = b_o - b_I$ (b_I is optional) with noise level h; the failure probability β.
Output: a child box b_c, `AGAIN`, or `LEAF`.

$\varepsilon_h \leftarrow \left(\frac{3}{4}\right)^{H-h} \cdot \varepsilon/100$
Draw $\zeta \sim \text{Lap}(1/\varepsilon_h)$. Let $\widetilde{R} \leftarrow \text{count}(c) + \zeta$.
if $\widetilde{R} < 370 \cdot \left(\log \frac{2\tilde{n}d}{\beta\delta} + \log\log u\right)/\varepsilon_h$ **then return** `AGAIN`
Draw $\gamma \sim \text{Lap}(1/\varepsilon_h)$. Let $\widetilde{T} \leftarrow \frac{2}{3}\text{count}(c) + \gamma$.
$b_c \leftarrow b_o$
repeat
  $b_l, b_r \leftarrow \text{split}(b_c)$
  if $\text{count}(b_l \cap c) > \text{count}(b_r \cap c)$ **then** $b_c \leftarrow b_l$ **else** $b_c \leftarrow b_r$
  if b_I exists **and** $b_c = b_I$ **then return** `LEAF`
  Draw $\eta \sim \text{Lap}(1/\varepsilon_h)$. Let $\tilde{\theta} \leftarrow \text{count}(b_c \cap c) + \eta$.
until $\tilde{\theta} < \widetilde{T}$ **or** $\text{size}(b_c) = 1$
Draw $\xi_l, \xi_r \sim \text{Lap}(1/\varepsilon_h)$.
if $\text{count}(b_l \cap c) + \xi_l > \text{count}(b_r \cap c) + \xi_r$ **then** $b_c \leftarrow b_l$ **else** $b_c \leftarrow b_r$
return b_c

Finally, after the BBD-tree has been constructed, we associate each cell c a noisy count, obtained by adding noise drawn from $\text{Lap}(40 \log \tilde{n}/\varepsilon)$ to $\text{count}(c)$. The following theorem guarantees its privacy and establishes some of its key properties that will be useful for proving its utility.

▶ **Theorem 7.** *The private BBD-tree preserves (ε, δ)-differential privacy. Furthermore, for $n > \frac{8}{\varepsilon}\log\frac{2}{\beta}$, with probability at least $1-\beta$, it has height $O(\log n)$ and $O(n)$ nodes, and every leaf cell c has either size$(c) = 1$ or count$(c) = O\left(\varepsilon^{-1}\cdot\left(\log\frac{n}{\beta\delta} + \log\log u\right)\right)$.*

Proof. Let $\ell_h = 10\cdot(\log\frac{2\tilde{n}d}{\beta\delta} + \log\log u)/\varepsilon_h$. Note that in a private shrink operation (Algorithm 2) on an input cell c with noise level h, all the noises are generated from $\mathrm{Lap}(1/\varepsilon_h)$. We say that a Laplace noise is *bounded* if its absolute value does not exceed ℓ_h.

We begin with the following lemma on one invocation of Algorithm 2:

▶ **Lemma 8.** *If all its Laplace noises are bounded, then Algorithm 2 (1) will not return* `LEAF`*; (2) if it returns* `AGAIN`*, then we have count$(c) \le 38\ell_h$; (3) otherwise we must have count$(c) \ge 36\ell_h$, and the algorithm returns a child box b_c which satisfies either of the following two properties: (a) size$(b_c) > 1$ and $\frac{1}{4}$count$(c) \le$ count$(b_c \cap c) \le \frac{3}{4}$count$(c)$; (b) size$(b_c) = 1$ and count$(b_c \cap c) \ge \frac{1}{4}$count$(c)$.*

Proof. (2) is easy to show. If Algorithm 2 returns `AGAIN`, since all noises are bounded, then it follows from line 3 that $\mathrm{count}(c) \le 37\ell_h - \zeta \le 38\ell_h$. The first part of (3) is also true, since if the algorithm does not return `AGAIN`, we must have $\mathrm{count}(c) \ge 37\ell_h - \zeta \ge 36\ell_h$.

Next, we prove the second part of (3). Let $b_c^{(1)}, b_l^{(1)}, b_r^{(1)}, \tilde{\theta}^{(1)}, \ldots, b_c^{(k)}, b_l^{(k)}, b_r^{(k)}, \tilde{\theta}^{(k)}$ denote the values of the variables $b_c, b_l, b_r, \tilde{\theta}$ at the end of each iteration in the repeat-until loop; note that $b_c^{(i)}$ is always the majority box between $b_l^{(i)}$ and $b_r^{(i)}$, $i = 1, \ldots, k$. Let b_c be the final output child box which is either $b_l^{(k)}$ or $b_r^{(k)}$ after a noisy comparison (line 13). We have $k \le d\log u$ as every d splits will decrease size$(b_c^{(i)})$ by a factor of 2. To prove the second part of (3), we only need to consider the case size$(b_c) > 1$. In this case, we have $\tilde{\theta}^{(k)} < \widetilde{T}$ and $\tilde{\theta}^{(k-1)} \ge \widetilde{T}$ according to line 11. Given $\tilde{\theta}^{(k)} < \widetilde{T}$, we have

$$\mathrm{count}(b_c \cap c) \le \mathrm{count}(b_c^{(k)} \cap c) \le \frac{2}{3}\mathrm{count}(c) - \eta^{(k)} + \gamma \le \frac{2}{3}\mathrm{count}(c) + 2\ell_h \le \frac{3}{4}\mathrm{count}(c),$$

where the first inequality is because b_c is either $b_l^{(k)}$ or $b_r^{(k)}$, while $b_c^{(k)}$ is the majority box between $b_l^{(k)}$ and $b_r^{(k)}$; the second inequality follows from the definition of $\tilde{\theta}^{(k)}$ and $\widetilde{T}$; the third inequality follows from the boundedness of $\eta^{(k)}$ and γ; and the fourth inequality is due to $\mathrm{count}(c) \ge 36\ell_h$. On the other hand, given $\tilde{\theta}^{(k-1)} > \widetilde{T}$, following similar arguments, we have

$$\mathrm{count}(b_c^{(k-1)} \cap c) \ge \frac{2}{3}\mathrm{count}(c) - \eta^{(k-1)} + \gamma \ge \frac{2}{3}\mathrm{count}(c) - 2\ell_h.$$

Assume w.l.o.g. that $b_c = b_l^{(k)}$. Then,

$$\begin{aligned}
2\cdot\mathrm{count}(b_c \cap c) &= \mathrm{count}(b_l^{(k)} \cap c) + \mathrm{count}(b_l^{(k)} \cap c)\\
&\ge \mathrm{count}(b_l^{(k)} \cap c) + (\mathrm{count}(b_r^{(k)} \cap c) - \xi_l + \xi_r)\\
&= \mathrm{count}(b_c^{(k-1)} \cap c) - \xi_l + \xi_r\\
&\ge \mathrm{count}(b_c^{(k-1)} \cap c) - 2\ell_h\\
&\ge \frac{2}{3}\mathrm{count}(c) - 4\ell_h,
\end{aligned}$$

where the first inequality follows from line 13, and the second inequality follows from the boundedness of ξ_l and ξ_r. Because $\mathrm{count}(c) \ge 36\ell_h$, we have

$$\mathrm{count}(b_c \cap c) \ge \frac{1}{3}\mathrm{count}(c) - 2\ell_h \ge \frac{1}{4}\mathrm{count}(c),$$

proving (3).

For (1), observe that $b_c^{(i)}$ is not identical to b_I (if it exists) for any $i \in [k]$, since

$$\text{count}(b_c^{(i)} \cap c) \geq \frac{1}{2}\text{count}(b_c^{(i-1)} \cap c) \geq \frac{1}{3}\text{count}(c) - \ell_h > 0,$$

where the first inequality is because $b_c^{(i)}$ is the majority box after a split on $b_c^{(i-1)}$; the second inequality follows from $\tilde{\theta}^{(i-1)} > \widetilde{T}$ and the boundedness of $\eta^{(i-1)}$ and γ; and the third inequality is due to $\text{count}(c) \geq 36\ell_h$. Thus the algorithm will not return `LEAF`. ◁

We are now ready to prove the utility properties of a private BBD-tree. If the event F that $n \leq \tilde{n} \leq 2n$ holds, the $O(\log n)$ height trivially follows from the construction algorithm. Due to the exponential tail of the Laplace distribution and $n > \frac{8}{\varepsilon}\log\frac{2}{\beta}$, we have $\Pr[F] \geq 1 - \beta/2$. To prove that the tree has $O(n)$ nodes and the property of the leaves stated in the theorem, consider the event E that the Laplace noises used in all invocations of Algorithm 2 when constructing the private BBD-tree are bounded, conditioned upon which the properties stated in Lemma 8 hold.

▶ **Lemma 9.** $\Pr[E] \geq 1 - \beta\delta/2$.

Proof. In one invocation of Algorithm 2 on a cell c, we draw $O(d \log u)$ independent Laplace noises from $\text{Lap}(1/\varepsilon_h)$, so they are all bounded with probability at least $1 - \beta\delta/(2\tilde{n}^{10})$, due to the exponential tail of Laplace distribution and a union bound. To bound the noise values in all the invocations, first consider the case when Algorithm 2 returns `AGAIN`. When this happens, we invoke it on c again with noise level $h \leftarrow h + 1$. This step is logically equivalent to making two copies of c, say c' and c'', such that c' is the only child of c and c'' is the only child of c'. Recall that we alternate between split and shrink operations, so we will later perform a shrink on c'' with $h \leftarrow h + 1$. Then in this logically equivalent BBD-tree, we only invoke one split operation or one shrink operation (or just make a copy of itself). Recall that we stop the construction when we reach $H = 2.5 \log \tilde{n}$, and one shrink operation may generate 4 levels of the tree (in case an interleaving split is needed), so the height of the logical tree is at most $4H = 10 \log \tilde{n}$. Thus, it has at most $\tilde{n}^{10}$ nodes, namely, at most $\tilde{n}^{10}$ invocations of Algorithm 2. Then the lemma is proved by a union bound. ◁

▶ **Lemma 10.** *Conditioned upon $E \wedge F$, the private BBD-tree has $O(n)$ nodes and every leaf cell c has either size$(c) = 1$ or count$(c) = O\left(\varepsilon^{-1} \cdot \left(\log \frac{n}{\beta\delta} + \log\log u\right)\right)$.*

Proof. Conditioned upon E, all the shrink operations have the three properties stated in Lemma 8. For any leaf c of size larger than 1, consider the path from root to c in the BBD-tree. There are exactly H invocations of Algorithm 2 on the path, as none of them has returned `LEAF`. At most $\log_{4/3} n < H$ (as the event F holds) of them have returned a child box, since every such shrink operation decreases the count by a factor at least $\frac{1}{4}$. This means that there is at least one invocation of Algorithm 2 that returned `AGAIN`.

Let c^* be the smallest cell on this path on which Algorithm 2 returned `AGAIN`, and let h^* be its noise level. By Lemma 8,

$$\text{count}(c^*) \leq 38\ell_{h^*} = O\left(\left(\frac{4}{3}\right)^{H-h^*} \cdot \left(\log\frac{nd}{\beta\delta} + \log\log u\right)/\varepsilon\right).$$

Each of the remaining $H - h^*$ shrink operations must have reduced the count by a factor of at least $\frac{1}{4}$, so

$$\text{count}(c) = O\left(\left(\frac{3}{4}\right)^{H-h^*} \cdot \ell_{h^*}\right) = O\left(\left(\log\frac{nd}{\beta\delta} + \log\log u\right)/\varepsilon\right).$$

To see that the BBD-tree has $O(n)$ nodes, consider the parent or grandparent, depending on which one is on the shrink level, of any leaf. Algorithm 2 might have been invoked on this node multiple times, but the last one must have returned a child box, implying that it contained at least ℓ_H data points by Lemma 8. Then we conclude that the private BBD-tree has $O(n)$ nodes since there are $O(\log n)$ levels and each level has at most $O(n/\ell_H)$ nodes. ◁

Finally, we prove its privacy.

▶ **Lemma 11.** *The private BBD-tree preserves (ε, δ)-differential privacy.*

Proof. Note that releasing $\tilde{n}$ preserves $(\varepsilon/4, 0)$-DP. Furthermore, there are two parts of the BBD-tree: its structure and the noisy counts associated with its cells. Since the height of the tree is at most $10 \log \tilde{n}$, the released noisy counts preserves $(\varepsilon/4, 0)$-DP by a standard argument. Below, we show that the tree structure is $(\varepsilon/2, \delta)$-DP. Then by the basic composition theorem, the private BBD-tree achieves (ε, δ)-DP.

The structure depends on the input point set P, as well as the Laplace noises generated internally during all the invocations of Algorithm 2. Let $\mathcal{M}(P, \Gamma)$ denote the tree structure constructed on point set P using noises Γ. We will show that for any two neighboring data sets P, P', and any tree structure y,

$$\mathsf{Pr}[\mathcal{M}(P, \Gamma) = y \wedge E] \leq e^{\varepsilon/2} \cdot \mathsf{Pr}[\mathcal{M}(P', \Gamma) = y], \tag{1}$$

where the probability is computed over the randomness of Γ, Then, for any set of output structures S, we have

$$\mathsf{Pr}[\mathcal{M}(P, \Gamma) \in S] \leq \mathsf{Pr}[\mathcal{M}(P, \Gamma) \in S \wedge E] + \mathsf{Pr}[\bar{E}] \leq e^{\varepsilon/2} \cdot \mathsf{Pr}[\mathcal{M}(P', \Gamma) \in S] + \delta,$$

namely, $\mathcal{M}$ is $(\varepsilon/2, \delta)$-DP.

To prove Eq. (1), it suffices to demonstrate an injection f from $\{\Gamma : \mathcal{M}(P, \Gamma) = y \wedge E\}$ to $\{\Gamma : \mathcal{M}(P', \Gamma) = y\}$ such that $\mathsf{Pr}(\Gamma) \leq e^{\varepsilon/2} \cdot \mathsf{Pr}(f(\Gamma))$. In order to achieve this, f can only change a small number of Laplace noises by a constant magnitude each. Note that the presence or absence of a data point will only affect the cells on a root-to-leaf path of y. For any Γ such that $\mathcal{M}(P, \Gamma) = y$ and all noises in Γ are bounded (i.e., event E holds), we follow the path starting from the root while making changes to Γ so that $\mathcal{M}(P', f(\Gamma)) = y$. Since the split operations do not depend on P, we only need to consider all the H shrink operations on that path.

For a cell c, we use $\mathrm{count}(c)$ to denote its count in P while $\mathrm{count}'(c)$ its count in P'. Note that $|\mathrm{count}(c) - \mathrm{count}'(c)| \leq 1$ for any c. We use ζ, η, $\widetilde{R}$, $\widetilde{T}$, ... to denote the noises in Γ, and ζ', η', $\widetilde{R}'$, $\widetilde{T}'$, ... for their counterparts in $f(\Gamma)$. The injection is defined as follows for each invocation of Algorithm 2:

1. Set $\zeta' := \zeta + \mathrm{count}(c) - \mathrm{count}'(c)$. Then we have $\widetilde{R}' = \widetilde{R}$, which makes the decisions at line 2 on P and P' are the same. Note that $|\zeta' - \zeta| = |\mathrm{count}(c) - \mathrm{count}'(c)| \leq 1$.
2. Recall the definitions of $b_c^{(\cdot)}, b_l^{(\cdot)}, b_r^{(\cdot)}, \eta^{(\cdot)}, \tilde{\theta}^{(\cdot)}$ in the proof of Lemma 8. Set $\widetilde{T}' := \widetilde{T} - 1$, ${\eta'}^{(i)} := \eta^{(i)}$ for $1 \leq i < k$, and ${\tilde{\theta}'}^{(k)} := \tilde{\theta}^{(k)} - 2$. Note that $|\gamma' - \gamma| \leq 2$ in order to achieve $\widetilde{T}' := \widetilde{T} - 1$ since $|\mathrm{count}(c) - \mathrm{count}'(c)| \leq 1$. We already have $\tilde{\theta}^{(i)} \geq \widetilde{T}$ for $1 \leq i < k$ and $\tilde{\theta}^{(k)} < \widetilde{T}$ on P. For $1 \leq i < k$, since $\tilde{\theta}^{(i)} \geq \widetilde{T}$, we have $\mathrm{count}(b_c^{(i)}) > \frac{2}{3}\mathrm{count}(c) - 2\ell_h > \frac{1}{2}\mathrm{count}(c) + \ell_h$ by $|\eta^{(i)}| \leq \ell_h$, $|\gamma| \leq \ell_h$ and $\mathrm{count}(c) \geq 36\ell_h$. This implies that the difference between $\mathrm{count}(b_l^{(i)})$ and $\mathrm{count}(b_r^{(i)})$ is at least ℓ_h, then the presence or absence of a single data point will not affect the choice of the majority

box $b_c^{(i)}$ on line 8, thus we have $|\tilde{\theta}'^{(i)} - \tilde{\theta}^{(i)}| \le 1$ by $|\text{count}(b_c^{(i)}) - \text{count}'(b_c^{(i)})| \le 1$ and $\eta'^{(i)} = \eta^{(i)}$. Since $b_c^{(k-1)}$ in the invocations on P and P' are the same, we can conclude that $|\text{count}'(b_c{}^{(k)}) - \text{count}(b_c^{(k)})| \le 1$, implying that $|\eta'^{(k)} - \eta^{(k)}| \le 3$ in order to achieve $\tilde{\theta}'^{(k)} := \tilde{\theta}^{(k)} - 2$. Given $\tilde{\theta}^{(i)} \ge \widetilde{T}$ for $1 \le i < k$ and $\tilde{\theta}^{(k)} < \widetilde{T}$, by the above mapping, we will have $\tilde{\theta}'^{(i)} \ge \widetilde{T}'$ for $1 \le i < k$ and $\tilde{\theta}'^{(k)} < \widetilde{T}'$. This implies that the repeat-until loop in the invocations on P and P' are the same.

3. Set $\xi_l' := \text{count}(b_l^{(k)}) + \xi_l - \text{count}'(b_l^{(k)})$, $\xi_r' := \text{count}(b_r^{(k)}) + \xi_r - \text{count}'(b_r^{(k)})$. This makes the decision on line 13 in the invocations on P and P' the same. We have $|\xi_l' - \xi_l| \le 1$ since $|\text{count}'(b_l^{(k)}) - \text{count}(b_l^{(k)})| \le 1$, and similarly $|\xi_r' - \xi_r| \le 1$.

Note that the injection defined above changes the values of the noises drawn from $\text{Lap}(1/\varepsilon_h)$ by at most 8, so the ratio between $\mathsf{Pr}(\Gamma)$ and $\mathsf{Pr}(f(\Gamma))$ is bounded by $\exp(8 \cdot \varepsilon_h)$ for the invocation of Algorithm 2 at noise level h. Across all H invocations, we have

$$\mathsf{Pr}(\Gamma) \le \exp\left(8 \cdot \sum_{h=0}^{H} \varepsilon_h\right) \cdot \mathsf{Pr}(f(\Gamma)) \le \exp(\varepsilon/2) \cdot \mathsf{Pr}(f(\Gamma)),$$

and Eq. (1) is proved. ◁

Then Theorem 7 follows from Lemma 9, 10, and 11. ◀

Now we analyze the error when using the private BBD-tree to answer approximate range queries.

▶ **Theorem 12.** *Given an (ε, δ)-differentially private BBD-tree and any α-approximate range query Q, for $n > \frac{8}{\varepsilon} \log \frac{2}{\beta}$, with probability at least $1 - \beta$, Algorithm 1 answers Q with an error of $O\left(\varepsilon^{-1} \cdot \left(\alpha^{-d}(\log \frac{n}{\beta\delta} + \log\log u) + \log^{1.5} n \log \frac{1}{\beta} + \alpha^{-d/2} \log n \log \frac{1}{\beta}\right)\right)$.*

4 Applications

4.1 Approximate nearest neighbor

In the *approximate nearest neighbor (ANN)* problem, the goal is to find a data point $p \in P$ such that $\text{dist}(p, q) \le (1 + \alpha)\text{dist}(p^*, q)$ for any given query point q, where p^* is the nearest neighbor of q in P and $0 < \alpha < 1$ is the approximation ratio. In the non-private setting, there is a standard approach using BBD-tree to solve ANN problem [2]. In the private setting, however, there are two differences. First, returning a data point in P will clearly breach privacy. Thus, we must settle for a slightly weaker target, namely, we will aim to return only the approximate distance to the NN, but not the NN itself. Second, the non-private BBD-tree has no errors, while its private versions do. This introduces some technical complications. In particular, we can no longer measure the approximation ratio of the ANN compared with the nearest neighbor p^*, but the τ-th nearest neighbor. More precisely, our goal is to return a distance r such that $r_{(1)} \le r \le (1 + \alpha)r_{(\tau)}$, where $r_{(\tau)}$ is the distance between q and its τ-th nearest neighbor in P. We call τ the *rank error*; obviously, smaller τ means better utility.

We present a more general algorithm for approximately finding the distance between a query point q and its k-th nearest neighbor in P, given an approximate range counting synopsis with error κ for $(\alpha/10)$-approximate spherical range queries. Note that since the whole space is $[u]^d$, the smallest and the largest possible distance between two distinct points are 1 and $\sqrt{d}u$ respectively. Let $B(q, w)$ denote the open ball with radius w centered at q. From the approximate range counting synopsis, we obtain approximate counts for

$|P \cap B(q, 1/2)|, |P \cap B(q, (1+\alpha/3)/2)|, |P \cap B(q, (1+\alpha/3)^2/2)|, \ldots, |P \cap B(q, (1+\alpha/3)^t/2)|$, where $t = \lceil \log_{1+\alpha/3}(2\sqrt{d}u) \rceil$; denote these approximate counts as $a_0, a_1, a_2, \ldots, a_t$. Let $i = \min\{i \mid a_i > k + \kappa\}$. Then we output $r = (1+\alpha/10)(1+\alpha/3)^i/2$ if $i \neq 0$; otherwise we return $r = 0$.

▶ **Lemma 13.** *For any $k \leq n - 2\kappa$, the above algorithm returns an r such that $r_{(k)} \leq r \leq (1+\alpha) r_{(k+2\kappa)}$.*

In the private setting, we use our private BBD-trees presented in Section 3.2 and 3.3 as the approximate range counting synopsis.

▶ **Theorem 14.** *There is an (ε, δ)-differentially private synopsis such that, with probability at least $1 - \beta$, the distance to the ANN for any query point can be found with a rank error of $O(\tau_{\varepsilon,\delta})$, where $t = \lceil \log_{1+\alpha/3}(2\sqrt{d}u) \rceil$, and*

$$\tau_{\varepsilon,\delta} = \begin{cases} \varepsilon^{-1} \alpha^{-d/2} \log u \log \frac{t}{\beta}, & \delta = 0; \\ \varepsilon^{-1} \cdot \left(\alpha^{-d} (\log \frac{n}{\beta\delta} + \log\log u) + \log^{1.5} n \log \frac{t}{\beta} + \alpha^{-d/2} \log n \log \frac{t}{\beta} \right), & \delta > 0. \end{cases}$$

4.2 k-NN classification

In k-NN classification, we are given m classes of points $P_1, P_2, \ldots, P_m$, and i is called the label of P_i. Given a query point q, the k-NN algorithm returns the label that is most common among its k nearest neighbors in $P = P_1 \cup \cdots \cup P_m$.

To obtain a private k-NN algorithm, we release an $(\varepsilon/2, \delta/2)$-private BBD-tree for P, as well as an $(\varepsilon/2, \delta/2)$-private BBD-tree for each P_i. This preserves an overall (ε, δ)-DP, since the presence or absence of one point affects only two private BBD-trees. For a given query point q, we invoke the algorithm in Section 4.1 on P with approximation factor $\alpha/3$, obtaining an r such that $r_{(k)} \leq r \leq (1+\alpha/3) r_{(k+O(\tau_{\varepsilon,\delta}))}$. Next, we query the private BBD-trees with approximation factor $\alpha/10$ to obtain approximate counts for $|P_1 \cap B(q, r)|, \ldots, |P_m \cap B(q, r)|$. Finally, we return the label i with the largest estimated $|P_i \cap B(q, r)|$.

Our private k-NN algorithm has the following utility guarantee. Intuitively, it says that as long as the correct label wins the majority in the k nearest neighbors of q by a poly-logarithmic margin, while the distances measured are allowed an $(1 \pm \alpha)$-factor approximation, then the algorithm will find it with high probability.

▶ **Theorem 15.** *There exists absolute constants C_1 and C_2 such that, with probability at least $1 - \beta$, our private k-NN synopsis can be used to find the correct label i for any query point q, provided that $|P_i \cap B(q, (1-\alpha) r_{(k)})| - |P_j \cap B(q, (1+\alpha) r_{(k+C_1 \cdot \tau_{\varepsilon,\delta})})| \geq C_2 \cdot \tau_{\varepsilon,\delta}$ for all $j \neq i$.*

Note that our algorithm releases a private k-NN classifier, which can be used to classify any query point. On the other hand, the private k-NN classifier in [11] only releases the classification results (with no utility guarantees) for a given set of queries, while no more queries can be accepted afterwards. Thus we solve a much more general problem than theirs.

5 Lower bounds

Observe that the errors in Theorem 6 and 14 for pure DP have an $O(\log u)$ term. In this section, we show that this is unavoidable even in one dimension. In particular, we prove the following lower bound on the one-dimensional ANN problem with a constant approximation factor $\alpha = 0.1$ and a constant failure probability $\beta = 1/3$ via a packing argument [13]. This in turn implies a lower bound for approximate range counting.

▶ **Theorem 16.** *Any $(\varepsilon, 0)$-differentially private one-dimensional ANN algorithm that, with probability at least $2/3$, returns an r such that $r_{(1)} \leq r \leq 1.1 \cdot r_{(\tau)}$ for every query point q, must have a rank error $\tau = \Omega\left(\frac{1}{\varepsilon} \log u\right)$.*

▶ **Corollary 17.** *Any $(\varepsilon, 0)$-differentially private one-dimensional approximate range counting algorithm that, with probability at least $2/3$, answers all 0.01-approximate range queries with error at most $\kappa_{\varepsilon,0}$, must have $\kappa_{\varepsilon,0} = \Omega\left(\frac{1}{\varepsilon} \log u\right)$.*

We can also show a lower bound under (ε, δ)-DP by a reduction from exact (i.e., no fuzziness) range counting [4]:

▶ **Theorem 18.** *Any (ε, δ)-differentially private one-dimensional approximate range counting algorithm that, with probability at least $2/3$, answers all 0.01-approximate range queries with error at most $\kappa_{\varepsilon,\delta}$, must have $\kappa_{\varepsilon,\delta} = \Omega\left(\frac{1}{\varepsilon} \cdot (\log^* u) \cdot \log(1/\delta)\right)$, for $\exp(-\varepsilon n/\log^* n) \leq \delta \leq 1/n^2$.*

Note that this lower bound means that (1) some dependency on u is inevitable even under (ε, δ)-DP, justifying the restriction that the points have integer coordinates; and (2) a logarithmic dependency on n/δ is also necessary.

References

1. Sunil Arya and David M Mount. Approximate range searching. In *Proceedings of the eleventh annual symposium on Computational geometry*, pages 172–181, 1995.
2. Sunil Arya, David M Mount, Nathan S Netanyahu, Ruth Silverman, and Angela Y Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM (JACM)*, 45(6):891–923, 1998.
3. Avrim Blum, Katrina Ligett, and Aaron Roth. A learning theory approach to non-interactive database privacy. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 609–618, 2008.
4. Mark Bun, Kobbi Nissim, Uri Stemmer, and Salil Vadhan. Differentially private release and learning of threshold functions. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 634–649. IEEE, 2015.
5. TH Hubert Chan, Elaine Shi, and Dawn Song. Private and continual release of statistics. In *International Colloquium on Automata, Languages, and Programming*, pages 405–417. Springer, 2010.
6. Graham Cormode, Cecilia Procopiuc, Divesh Srivastava, Entong Shen, and Ting Yu. Differentially private spatial decompositions. In *2012 IEEE 28th International Conference on Data Engineering*, pages 20–31. IEEE, 2012.
7. Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.
8. Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N Rothblum. Differential privacy under continual observation. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 715–724, 2010.
9. Cynthia Dwork, Moni Naor, Omer Reingold, and Guy N Rothblum. Pure differential privacy for rectangle queries via private partitions. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 735–751. Springer, 2015.
10. Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.
11. Mehmet Emre Gursoy, Ali Inan, Mehmet Ercan Nergiz, and Yucel Saygin. Differentially private nearest neighbor classification. *Data Mining and Knowledge Discovery*, 31(5):1544–1575, 2017.

12 Moritz Hardt and Guy N. Rothblum. A multiplicative weights mechanism for privacy-preserving data analysis. In *2010 IEEE Annual Symposium on Foundations of Computer Science*, pages 61–70. IEEE, 2010.

13 Moritz Hardt and Kunal Talwar. On the geometry of differential privacy. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 705–714, 2010.

14 Michael Hay, Vibhor Rastogi, Gerome Miklau, and Dan Suciu. Boosting the accuracy of differentially private histograms through consistency. *Proceedings of the VLDB Endowment*, 3(1-2):1021–1032, 2010.

15 Ziyue Huang and Ke Yi. Approximate range counting under differential privacy. Available at `https://www.cse.ust.hk/~yike/DPRangeCount.pdf`.

16 Chao Li, Michael Hay, Gerome Miklau, and Yue Wang. A data-and workload-aware algorithm for range queries under differential privacy. *Proceedings of the VLDB Endowment*, 7(5):341–352, 2014.

17 J. Matoušek. *Geometric Discrepancy: An Illustrated Guide*. Springer-Verlag, Berlin, 1999.

18 Shanmugavelayutham Muthukrishnan and Aleksandar Nikolov. Optimal private halfspace counting via discrepancy. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 1285–1292, 2012.

19 Aleksandar Nikolov, Kunal Talwar, and Li Zhang. The geometry of differential privacy: the sparse and approximate cases. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 351–360, 2013.

20 Wahbeh Qardaji, Weining Yang, and Ninghui Li. Differentially private grids for geospatial data. In *2013 IEEE 29th international conference on data engineering (ICDE)*, pages 757–768. IEEE, 2013.

21 Wahbeh Qardaji, Weining Yang, and Ninghui Li. Understanding hierarchical methods for differentially private histograms. *Proceedings of the VLDB Endowment*, 6(14):1954–1965, 2013.

22 Salil Vadhan. The complexity of differential privacy. In *Tutorials on the Foundations of Cryptography*, pages 347–450. Springer, 2017.

23 Xiaokui Xiao, Guozhang Wang, and Johannes Gehrke. Differential privacy via wavelet transforms. *IEEE Transactions on knowledge and data engineering*, 23(8):1200–1214, 2010.

24 Jun Zhang, Xiaokui Xiao, and Xing Xie. Privtree: A differentially private algorithm for hierarchical decompositions. In *Proceedings of the 2016 International Conference on Management of Data*, pages 155–170, 2016.

Sublinear Average-Case Shortest Paths in Weighted Unit-Disk Graphs

Adam Karczmarz
Institute of Informatics, University of Warsaw, Poland

Jakub Pawlewicz
Institute of Informatics, University of Warsaw, Poland

Piotr Sankowski
Institute of Informatics, University of Warsaw, Poland

Abstract

We consider the problem of computing shortest paths in weighted unit-disk graphs in constant dimension d. Although the single-source and all-pairs variants of this problem are well-studied in the plane case, no non-trivial exact distance oracles for unit-disk graphs have been known to date, even for $d = 2$.

The classical result of Sedgewick and Vitter [Algorithmica '86] shows that for weighted unit-disk graphs in the plane the A^* search has average-case performance superior to that of a standard shortest path algorithm, e.g., Dijkstra's algorithm. Specifically, if the n corresponding points of a weighted unit-disk graph G are picked from a unit square uniformly at random, and the connectivity radius is $r \in (0, 1)$, A^* finds a shortest path in G in $O(n)$ expected time when $r = \Omega(\sqrt{\log n/n})$, even though G has $\Theta((nr)^2)$ edges in expectation. In other words, the work done by the algorithm is in expectation proportional to the number of vertices and not the number of edges.

In this paper, we break this natural barrier and show even stronger sublinear time results. We propose a new heuristic approach to computing point-to-point exact shortest paths in unit-disk graphs. We analyze the average-case behavior of our heuristic using the same random graph model as used by Sedgewick and Vitter and prove it superior to A^*. Specifically, we show that, if we are able to report the set of all k points of G from an arbitrary rectangular region of the plane in $O(k + t(n))$ time, then a shortest path between arbitrary two points of such a random graph on the plane can be found in $O(1/r^2 + t(n))$ expected time. In particular, the state-of-the-art range reporting data structures imply a sublinear expected bound for all $r = \Omega(\sqrt{\log n/n})$ and $O(\sqrt{n})$ expected bound for $r = \Omega(n^{-1/4})$ after only near-linear preprocessing of the point set.

Our approach naturally generalizes to higher dimensions $d \geq 3$ and yields sublinear expected bounds for all $d = O(1)$ and sufficiently large r.

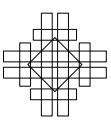
2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases unit-disk graphs, shortest paths, distance oracles

Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.46

Related Version The full version of this paper also describes a slightly different query algorithm whose running time has a better dependence on n for higher dimensions $d > 2$.
Full Version: `https://arxiv.org/abs/2103.09684`

Funding *Adam Karczmarz*: supported by ERC Consolidator Grant 772346 TUgbOAT and by the Foundation for Polish Science (FNP) via the START programme.
Jakub Pawlewicz: supported by ERC Consolidator Grant 772346 TUgbOAT.
Piotr Sankowski: supported by ERC Consolidator Grant 772346 TUgbOAT.



1 Introduction

Computing shortest paths is certainly one of the most fundamental graph problems and has numerous theoretical and practical applications. The two classical variants of the shortest paths problem are *single-source shortest paths* (SSSP) and *all-pairs shortest path* (APSP). A common generalization of these variants is the *distance oracle* problem, where we are allowed to preprocess a given network into a (possibly small) data structure that is later used to answer arbitrary *point-to-point* shortest paths queries. Clearly, SSSP and APSP algorithms can be viewed as extreme solutions to the distance oracle problem: the former can be used without any preprocessing to query a distance in near-linear time, whereas the latter precomputes the answers to all the n^2 possible queries and thus can answer queries in constant time. Hence, when constructing distance oracles we seek a tradeoff between these two extremes. Unfortunately, it is not known how to obtain a non-trivial (with both subquadratic space and sublinear query time) *exact* distance oracle for general graphs. Subquadratic space and constant query time oracle is only known for undirected weighted graphs if approximation factor of at least 3 is allowed [19].

Due to this theoretical inefficiency of distance oracles, researchers either focus on special graph classes, or study approximate approaches. On one hand, near-optimal (in terms of space and query time) exact distance oracles have been recently described for planar graphs [8]. On the other hand, for many important graph classes near-optimal $(1+\epsilon)$-approximate distance oracles are known [15, 2].

Nevertheless, in practice heuristic approaches are usually preferable – for an overview of used techniques see [3]. However, the term "heuristic" in this domain usually refers to ways of speeding up exact algorithms. There are some examples of heuristics that have been analyzed theoretically and proved to yield speedups in meaningful settings, see e.g., [1].

Perhaps the most well-known heuristic approach to speeding up a shortest path computation is a variant of Dijkstra's algorithm called the A^* *search* [13]. This algorithm incorporates a heuristic function that lower-bounds the distance to the target and uses it to decide which search paths to follow first. The algorithm is still guaranteed to find the shortest path to the target vertex, but the number of vertices explored can be much smaller compared to the standard Dijkstra's algorithm. For example, if vertices of the network correspond to points in the plane, the Euclidean distance to the target is a valid and well-working heuristic function. The natural question arises when such algorithms perform provably better than in the worst-case.

1.1 Shortest paths in weighted unit-disk graphs

The seminal result that answers this question is by Sedgewick and Vitter [18], who studied the performance of A^* search on various random geometric graph models. Perhaps the most interesting of their results concerns the *weighted unit-disk graphs*. In a weighted unit-disk graph with *connectivity radius* r, vertices correspond to points on the plane. An edge between two distinct vertices (points) u, v exists in such a graph if $||u-v||_2 \leq r$ and has weight $||u-v||_2$. This class of geometric graphs has been widely studied from the algorithmic perspective since such graphs can model e.g., ad-hoc communication networks. A *random weighted unit-disk graph* G, given n and radius $r \in (0,1)$, is obtained from a set of n random points of a unit square $[0,1]^2$. Note that such a random G has $\Theta((nr)^2)$ edges in expectation. However, Sedgewick and Vitter [18] show that, assuming that the neighbors of each vertex in G are stored explicitly, one can compute a point-to-point shortest path in G using A^*

search in $O(n)$ expected time, i.e., independent of r and sublinear in the size of the edge set of G. In other words, they have given an exact distance oracle for random weighted unit-disk graphs that in expectation requires $O((nr)^2)$ space and $O(n)$ query time.

Sedgewick and Vitter's result [18] can be also interpreted as follows: for weighted unit-disk graphs $G = (V, E)$, just storing the graph explicitly allows $O(n)$-time queries for an average-case graph G. Whereas such a query time is sublinear in the graph size, the $\Theta((nr)^2)$ space used might be *superlinear* in the graph's description – observe that a weighted unit-disk graph can be described using $O(n)$ space solely with n point locations and the connectivity radius r. In recent years efficient single-source shortest paths algorithm for weighted unit-disk graphs have been proposed [4, 14, 20], culminating in the $O(n \log^2 n)$ algorithm of Wang and Xue [20]. Note that their worst-case bound is near-optimal and almost matches the bound of [18] which holds only on average. All-pairs shortest paths in weighted unit-disk graphs can be computed slightly faster that running single-source computations n times [6]. To the best of our knowledge, no exact distance oracle with non-obvious space and query bounds for this graph class is known. On the contrary, a very efficient $(1+\epsilon)$-approximate distance oracle with near-optimal space, preprocessing, and query bounds was given by Chan and Skrepetos [7].

The notion of a weighted unit-disk graph naturally generalizes to three- and higher dimensions: an edge between two vertices appears if the d-dimensional balls of radius r at these points intersect. We are not aware of any non-trivial results on computing shortest paths in such graphs for $d \geq 3$.

1.2 Our results

Observe that all of the above algorithms in order to answer distance queries require work essentially proportional to the number of vertices and not the number of edges. In this paper, we break this natural barrier and show an even stronger sublinear time results.

We propose a natural heuristic approach to computing exact shortest paths in weighted unit-disk graphs. Following Sedgewick and Vitter, we analyze its average-case query time by studying its performance on a random n-vertex graph with connectivity radius r in the unit square $[0,1]^2$, where $r = \Omega\left(\sqrt{\log(n)/n}\right)$.[1] In this setting, we prove that after near-linear preprocessing, the query procedure of our average-case distance oracle has $O(1/r^2 + \sqrt{n})$ expected running time. Formally, we prove:

▶ **Theorem 1.** *Let $r \in (0,1)$ be such that $r = \Omega\left(\sqrt{\log(n)/n}\right)$. Let G be a weighted unit-disk graph with connectivity radius r on a set P of n points picked uniformly at random from the unit square $[0,1]^2$. Let $\mathcal{D}$ be a data structure that, after preprocessing P in $O(p(n))$ time, supports reporting all k points in P lying in an arbitrary (not necessarily orthogonal) rectangular subregion in $O(k + t(n))$ time. Then, there exists an exact distance oracle on G with $O(p(n))$ preprocessing time and $O(1/r^2 + t(n))$ expected query time.*

The state-of-the-art range searching data structures [5] imply that $t(n) = O(\sqrt{n})$ using $O(n)$ space and $O(n \log n)$ preprocessing. Consequently, for $r = \Omega(1/n^{1/4})$ the expected query time is $O(\sqrt{n})$ and it remains truly sublinear for all $r = \Omega(\sqrt{\log(n)/n})$ – improving the running time of Sedgewick and Vitter in the full range of parameters r they consider.

[1] This simplifying assumption has also been made by Sedgewick and Vitter [18] and excludes only very sparse graphs with $m = O(n \log n)$ from our consideration. Moreover, it is known that if $r = o(\sqrt{\log(n)/n})$, then the random unit-disk graph is disconnected with high probability [12].

The general idea behind our heuristic algorithm for computing a shortest $s-t$ path is fairly intuitive: we run the single-source shortest paths algorithm limited to increasingly "fat" rectangular subregions of G surrounding the segment $s-t$. The subregions of interest are computed using a range reporting data structure which constitutes the only preprocessed information of our oracle. Since dynamic variants of such range searching data structures are known [16] (with query and space bound matched up to polylogarithmic factors, and polylogarithmic update bounds), our heuristic distance oracle can be trivially dynamized as well (see Remark 12).

Another advantage of our algorithm is that it easily generalizes to higher dimensions. Using new ideas we prove that for random weighted unit-disk graphs[2] in $[0,1]^d$, the expected query time is $O(\min(1/r^{2d-1}, n) + t_d(n))$, assuming one can report the points from an arbitrary (not necessarily orthogonally aligned) d-dimensional hyperrectangle in $O(t_d(n)+k)$ time. It is known [5] that $t_d(n) = O(n^{1-1/d})$ so this expected time is sublinear in n unless $r = \Omega\left(n^{-\frac{1}{2d-1}}\right)$. It is worth noting that for $d=2$, the expected query time has a "better" dependence, i.e., $O(1/r^d)$, on r than for $d \geq 3$ where the dependence is $O(1/r^{2d-1})$. This is justified by the fact that whereas single-source shortest paths in weighted unit-disk graphs for $d=2$ can be computed nearly-optimally [20], no non-trivial algorithm like this is known for $d \geq 3$ and we have to resort to running the standard Dijkstra's algorithm.

Undoubtedly, the technical difficulty of our result lies in the probabilistic analysis. We use similar approach to the one used by Sedgewick and Vitter [18] to bound the probability that the sought path exist in ellipsoidal grid-like regions called *channels*. However, in order to avoid looking at all the edges incident to a vertex we need to use a new heuristic that allows us to consider only edges induced within an rectangular region.

Interestingly, we also identify a shortcoming in their original analysis for the two-dimensional case and give a more delicate argument inspired by the techniques from so-called *oriented percolation theory* (see e.g., [10]). The original result of Sedgewick and Vitter [18] wrongly limited the sets of directed paths going through the channel grid. Thus the resulting probability that a path exists was overestimated. The more detailed description of the shortcoming of the original proof can be found in the full version of this paper.

We note that for $d=2$ the graph model considered here has been widely studied in the context of wireless networks [12]. For example, Gupta and Kumar [11] studied the connectivity of such networks, and have shown a critical r above which the graph is connected with high probability. This result was generalized by Penrose [17] to k-connectivity. Our result gives the first known sublinear shortest path routing oracle for such networks. In a sense, our results call for further work on exact distance oracles for weighted unit disk graphs. In particular it might suggest that near-linear space and sublinear query time exact distance oracles in the worst-case exist, as proving such result over random graphs can be a seen as a proof-of-concept for such a possibility.

2 Preliminaries

A *weighted unit-disk graph* $G=(V,E)$ with *connectivity radius* r is an undirected geometric graph whose vertices are identified with some n points in $\mathbb{R}^d$, where $d \geq 2$ is a constant. The edge set of G contains an edge $\{u,v\}$ for all $u=(u_1,\ldots,u_d)$, $v=(v_1,\ldots,v_d) \in V$ such that $||u-v||_2 = \sqrt{\sum_{i=1}^{d}(u_i-v_i)^2} \leq r$. For brevity, in the following we omit the subscript and write $||x-y||$ instead of $||x-y||_2$.

[2] Since a disk is a subset of a plane, in higher dimensions $d>2$, it would be perhaps more appropriate to call such graphs *weighted unit-ball graphs*. However, anyway, we stick to the well-established term *weighted unit-disk graph* since our main result concerns the plane case $d=2$.

For $u, v \in V$, by $\mathrm{dist}_G(u, v)$ we denote the length of a shortest $u \to v$ path in G.

We consider *exact distance oracles* for weighted unit-disk graphs G, i.e., data structures that preprocess G (ideally into a near-linear space data structure using near-linear time) and then accept point-to-point distance queries, i.e., given query vertices $u, v \in V$, compute $\mathrm{dist}_G(u, v)$. The algorithms we propose can be straightforwardly extended to also report actual shortest paths within the same asymptotic time bound. Hence, we focus only on computing distances.

In order to perform a meaningful average-case analysis of a distance oracle's query algorithm on weighted unit-disk graphs for a given r, we need to limit the space of possible graphs. To this end, following Sedgewick and Vitter [18], for $r \in (0, 1)$ we limit our attention to graphs with all n points in $[0, 1]^d$. In order to compute the average running time of a shortest path query, we would like to compute it over all possible such graphs. Equivalently, we study the expected running time of a query algorithm on a *random graph* G, where each of n points is picked uniformly at random from $[0, 1]^d$. Note that in such a case, each vertex w has $\Theta(nr^d)$ neighbors in expectation: the probability that another vertex z is connected with w with an edge equals the probability that z is picked in the d-dimensional ball of radius r around v which clearly has volume $\Theta(r^d)$.

We also assume $r \geq \left(\frac{\beta \log n}{n}\right)^{1/d}$ for a sufficiently large constant $\beta > 1$. Then, the random graph G has $\Omega(n \log n)$ edges in expectation. For $d = 2$, the bound $r = \Omega\left(\left(\frac{\log n}{n}\right)^{1/d}\right)$ has also been assumed by Sedgewick and Vitter [18], as it greatly simplifies calculations. Moreover, for $r = o\left(\left(\frac{\log n}{n}\right)^{1/d}\right)$, with high probability G is not connected [11].

3 The distance oracle

3.1 Preprocessing

Let the coordinates of the n points of an input weighted unit-disk graph G be given. In the preprocessing phase, in $O(n \log n)$ time we build a simplex range searching data structure on V [5]. This data structure requires only linear space and allows $O(n^{1-1/d} + k)$ worst-case time queries reporting all of the k input points in an arbitrary hyperrectangle (with sides not necessarily parallel to the axes) of $\mathbb{R}^d$.

3.2 Query algorithm

Suppose the query is to compute $\mathrm{dist}_G(s, t)$ for $s, t \in V$. Let

$$w = ||t - s||.$$

Clearly, we have $\mathrm{dist}_G(s, t) \geq w$. Moreover, in the following we assume $w > r$, since otherwise we trivially have $\mathrm{dist}_G(s, t) = w$.

Let us first move and rotate the coordinate system so that the origin is now in s and the direction of the first axis is the same as $\overrightarrow{st}$, thus we have $s = (0, 0, \ldots, 0)$ and $t = (w, 0, \ldots, 0)$ in the new coordinate system.

▶ **Observation 2.** *Let $W \geq w$ denote an upper bound on $\mathrm{dist}(s, t)$. If a s–t shortest path in G contains a vertex $x \in V$ then*

$$||x - s|| + ||x - t|| \leq W. \tag{1}$$

Inequality (1) describes a set of points contained in a d-dimensional ellipsoid. The first axis of that ellipsoid has length $W/2$, whereas all other $d-1$ axes have length R, where R satisfies $(w/2)^2 + R^2 = (W/2)^2$. Hence:

$$R = \frac{1}{2}\sqrt{W^2 - w^2}.$$

Note that the ellipsoid is contained in a d-dimensional *bounding box*

$$\left[-\frac{W-w}{2}, \frac{W+w}{2}\right] \times [-R, R] \times \ldots \times [-R, R] \tag{2}$$

with first side length equal to W and the other $d-1$ side lengths equal to $2R$.

We will later pick an unbounded increasing function $W_{\mathrm{ub}} : \mathbb{Z}_+ \to \mathbb{R}_+$ with values depending on n, d, r, with the goal of defining increasingly large bounding boxes, as follows.

▶ **Definition 3.** *For a given integer $i \geq 1$, by $\mathrm{BE}(i)$ we denote the set of points satisfying inequality (1) for $W = W_{\mathrm{ub}}(i)$. Similarly, by $\mathrm{BB}(i)$ we denote the bounding box as in formula (2) for $W = W_{\mathrm{ub}}(i)$.*

Our entire algorithm will be to run a single-source shortest paths algorithm on the graphs

$$G(i) = (V_i, E_i) = G \cap \mathrm{BB}(i),$$

subsequently for $i = 1, 2, \ldots, i_{\max}$ (where $i_{\max}$ is to be set later) until an $s \to t$ path of length no more than $W_{\mathrm{ub}}(i)$ is found. If we are successful with that for some i, the found path is returned as the shortest $s \to t$ path. Otherwise, we simply run Dijkstra's algorithm from s on the entire G and either return the found shortest $s \to t$ path, or return ∞ if no path is found.

▶ **Lemma 4.** *The above algorithm is correct.*

Proof. The algorithm clearly stops. Moreover, the final Dijkstra step ensures that an $s \to t$ path is found if and only if a $s \to t$ path in G exists.

To prove correctness suppose that $\mathrm{dist}_G(s,t) < \infty$. Let i^* be the first i for which $\mathrm{dist}_{G(i^*)}(s,t) \leq W_{\mathrm{ub}}(i^*)$, if such i^* exists. Since $G(i^*) \subseteq G$, $\mathrm{dist}_G(s,t) \leq \mathrm{dist}_{G(i^*)}(s,t)$ and hence $\mathrm{dist}_G(s,t) \leq W_{\mathrm{ub}}(i^*)$. So, by Observation 2, a path of length $\mathrm{dist}_G(s,t)$ has all its vertices in $\mathrm{BE}(i^*) \subseteq \mathrm{BB}(i^*)$. This proves $\mathrm{dist}_G(s,t) \geq \mathrm{dist}_{G(i^*)}(s,t)$, so in fact $\mathrm{dist}_G(s,t) = \mathrm{dist}_{G(i^*)}(s,t)$.

If i^* does not exists, we run Dijkstra's algorithm on the entire graph G, so clearly a shortest $s \to t$ path is returned. ◀

Let $\mathrm{T}^V_{\mathrm{gen}}(i)$ and $\mathrm{T}^E_{\mathrm{gen}}(i)$ be the times required to find sets V_i and E_i, respectively. Since V_i is defined as a subset of V inside a d-dimensional bounding box $\mathrm{BB}(i)$, it can be clearly computed using a single query to the preprocessed range searching data structure. Hence,

$$\mathrm{T}^V_{\mathrm{gen}}(i) = O(n^{1-1/d} + |V_i|).$$

Denote by $T_d(i)$ the worst-case running time of step i. The cost $T_d(i)$ might differ depending on the algorithm that we use to find a shortest path in $G(i)$. Note that $G(i)$ is a weighted unit-disk graph, so if $d = 2$, and we employ the recent nearly-linear (in the number of vertices), albeit difficult to implement, algorithm of Wang and Xue [20], so we have:[3]

$$T_2(i) = O\left(|V_i| \log^2(|V_i| + 2) + \mathrm{T}^V_{\mathrm{gen}}(i)\right) = O\left(|V_i| \log^2(|V_i| + 2) + \sqrt{n}\right). \tag{3}$$

[3] We use $\log(|V_i| + 2)$ instead of just $\log |V_i|$ to make sure this term is at least a positive constant.

On the other hand, if $d > 2$, we need to use the simple-minded Dijkstra's algorithm to find a shortest path in $G(i)$, so we have

$$T_d(i) = O\left(|V_i| \log(|V_i| + 2) + |E_i| + \mathrm{T}^E_{\mathrm{gen}}(i)\right). \tag{4}$$

Let $\bar{P}(i)$ be the probability that we fail to find a path of length at most $W_{\mathrm{ub}}(i)$ in the graph G_i. The expected running time of the algorithm is then

$$O\left(\sum_{i=1}^{i_{\max}} \bar{P}(i-1) \cdot \mathbb{E}[T_d(i)] + \bar{P}(i_{\max}) \cdot n^2\right). \tag{5}$$

We will prove that by choosing

$$i_{\max} = \Theta(nr^d), \tag{6}$$

and

$$W_{\mathrm{ub}}(i) = \Theta\left(w \cdot \sqrt{1 + \left(\frac{i}{nr^d}\right)^{\frac{2}{d-1}}}\right) = O(w), \tag{7}$$

as described precisely in Section 5, we can obtain the following key bound. The proof of this bound is covered in Sections 4 and 5.

▶ **Lemma 5.** *For $i = 1, \ldots, i_{max}$, $\bar{P}(i) \le e^{-i}$.*

We now derive bounds on the expected sizes of sets V_i and E_i.

▶ **Lemma 6.** *For $i = 1, \ldots, i_{max}$, $\mathbb{E}[|V_i|] = \Theta\left((w/r)^d i\right)$.*

Proof. Clearly, $\mathbb{E}[|V_i|]$ equals the volume of $\mathrm{BB}(i)$ times n. For $W = W_{\mathrm{ub}}(i)$ we have

$$R = \frac{1}{2}\sqrt{W^2 - w^2} = \Theta\left(w \cdot \left(\frac{i}{nr^d}\right)^{\frac{1}{d-1}}\right). \tag{8}$$

Since $\mathrm{BB}(i)$ has size $W \times 2R \times \ldots \times 2R$, its volume is

$$W \cdot (2R)^{d-1} = \Theta(w) \cdot \Theta(R^{d-1}) = \Theta(w) \cdot \Theta\left(\frac{w^{d-1} i}{nr^d}\right) = \Theta\left(\frac{1}{n} \cdot \left(\frac{w}{r}\right)^d \cdot i\right). \qquad \blacktriangleleft$$

In order to analyse the running time we will need the following technical lemma whose proof can be found in the full version.

▶ **Lemma 7.** *Let X be a random variable from a binomial distribution with n variables and mean $\mathbb{E}[X] = \mu = \Omega(1)$. Then for any constant integer $\alpha \ge 1$:*

$$\mathbb{E}[X \cdot \log^\alpha(X + 2)] = O(\mathbb{E}[X] \cdot \log^\alpha(\mathbb{E}[X] + 2))) = O(\mu \cdot \log^\alpha(\mu + 2))).$$

▶ **Corollary 8.** *For any integer $\alpha \ge 1$ we have*

$$\mathbb{E}[|V_i| \log^\alpha(|V_i| + 2)] = O(\mathbb{E}[|V_i|] \cdot \log^\alpha(\mathbb{E}[|V_i|] + 2)).$$

Proof. We can apply Lemma 7 since $\mathbb{E}[|V_i|] = \Omega(1)$ by Lemma 6. ◀

▶ **Lemma 9.** *Let*

$$f^E(i) = \min\{nr^d, (w/r)^{d-1}i\} = \begin{cases} (w/r)^{d-1}i & \text{for } r \geq \sqrt[2d-1]{w^{d-1}i/n} \\ nr^d & \text{otherwise.} \end{cases}$$

Then for $i = 1, \ldots, i_{max}$, $\mathbb{E}[|E_i|] = \mathbb{E}[|V_i|] \cdot O(f^E(i))$.

Proof. Take a vertex $v \in V_i$. All neighbours of v in $G(i)$ belong to the intersection of the d-dimensional ball of radius r centered at v, and the bounding box $\mathrm{BB}(i)$. This intersection, on one hand, is contained in a box of size $2r \times 2R \times \cdots \times 2R$, where $R = \frac{1}{2}\sqrt{(W_{\mathrm{ub}}(i))^2 - w^2}$ (see (8)). On the other hand, it is trivially inside a ball of radius r. In the former case the volume of the box with v's neighbours is

$$O(rR^{d-1}) = O\left(\frac{1}{n}(w/r)^{d-1} \cdot i\right)$$

In the latter case the volume is $O(r^d)$. Therefore, the expected number of neighbours of v is

$$O\left(n \cdot \min\left\{\frac{1}{n}(w/r)^{d-1} \cdot i, r^d\right\}\right) = O(\min\{(w/r)^{d-1} \cdot i, nr^d\}). \tag{9}$$

By linearity of expectation we get the desired bound on $\mathbb{E}[|E_i|]$. ◀

The following lemma describes how to efficiently generate the edges E_i when we use Dijkstra's algorithm (for $d \geq 3$).

▶ **Lemma 10.** *Let* f^E *be as in Lemma 9. Given* V_i*, the edge set* E_i *can be computed in* $T^E_{gen}(i) = O(\mathbb{E}[|V_i|] \cdot f^E(i))$ *expected time.*

Proof. We divide $[0,1]^d$ into cubes of size $r \times r \times \cdots \times r$. With each non-empty cube we will keep a list of vertices from V_i that belongs to that cube. We build these lists by iterating over all $v \in V_i$ and assign v to the appropriate cube's list. Technically speaking, the lists are stored in a hash table with expected $O(1)$ insertion and access time (see e.g., [9]): note that the cubes can be mapped to integers $[1, (\lceil 1/r \rceil)^d]$ and we have $(\lceil 1/r \rceil)^d = O(n)$ by $r = \Omega\left((\log(n)/n)^{1/d}\right)$. To find the edges, for each v we iterate over all vertices w belonging to the same cube as v or a neighbouring cube and check whether $||v - w|| \leq r$. There are at most 3^d such cubes and each neighbor of v necessarily lies in these neighboring cubes.

Each cube contains $O(n \min\{rR^{d-1}, r^d\})$ vertices in expectation, where we again set $R = \frac{1}{2}\sqrt{(W_{\mathrm{ub}}(i))^2 - w^2}$ (see (8)). Recall from (9) in Lemma 9 that this quantity is $O(f^{\mathrm{E}}(i))$. This is because if $2R < r$ then the cube's intersection with $\mathrm{BB}(i)$ has size at most $r \times (2R) \times \cdots \times (2R)$ and only in that part of the cube the vertices from V_i can appear. Therefore, the expected total work for each vertex will be $O(3^d \cdot f^{\mathrm{E}}(i)) = O(f^{\mathrm{E}}(i))$. Thus, by linearity of expectation, the expected running time is indeed $O(\mathbb{E}[|V_i|] \cdot f^E(i))$. ◀

We are now ready to prove the following theorem bounding the expected running time of the query algorithm.

▶ **Theorem 11.** *The expected running time of the query algorithm on an* n*-vertex random weighted unit-disk graph in* $[0,1]^d$ *with connectivity radius* r *is*

(a) $O\left((w/r)^2 \log^2(1 + w/r) + \sqrt{n}\right)$ *for* $d = 2$,
(b) $O\left((w/r)^{2d-1} + n^{1-1/d}\right)$ *for* $d \geq 3$ *and* $r \geq \sqrt[2d-1]{w^{d-1}/n}$,
(c) $O(nw^d + n^{1-1/d})$ *otherwise.*

Proof. In all cases we will bound the expected query time as given in sum (5):

$$O\left(\sum_{i=1}^{i_{\max}} \bar{P}(i-1)\cdot \mathbb{E}[T_d(i)] + \bar{P}(i_{\max})\cdot n^2\right).$$

First of all, note that by Equation 6, Lemma 5 and the assumption $r \geq (\beta \log(n)/n)^{1/d}$ where $\beta > 1$ is a large enough constant, for some constant $\gamma > 0$ we have:

$$i_{\max} \geq \gamma \cdot nr^d \geq \gamma \cdot \beta \log n$$

So, picking $\beta = 2/\gamma$ gives us

$$\bar{P}(i_{\max})\cdot n^2 = O\left(e^{-i_{\max}}\cdot n^2\right) = O\left(e^{-2\log n}\cdot n^2\right) = O(1).$$

Hence, we can focus on the below sum. By Lemma 5, we have:

$$O\left(\sum_{i=1}^{i_{\max}} \bar{P}(i-1)\cdot \mathbb{E}[T_d(i)]\right) = O\left(\sum_{i=1}^{\infty} \mathbb{E}[T_d(i)]e^{-(i-1)}\right) = O\left(\sum_{i=1}^{\infty} \mathbb{E}[T_d(i)]e^{-i}\right).$$

In the following, we will use the asymptotic formula $\sum_{i=1}^{\infty} f(i)e^{-i} = O(1)$ that holds for any function $f(i) = \mathrm{poly}(i)$. Recall that $w > r$.

Let us first prove item (a). By (3) and Lemma 6, we have:

$$\begin{aligned}
&O\left(\sum_{i=1}^{\infty} \mathbb{E}[T_2(i)]e^{-i}\right)\\
&= O\left(\sum_{i=1}^{\infty} (w/r)^2\cdot i\cdot \log^2\left((w/r)^2 i + 2\right)\cdot e^{-i} + \sum_{i=1}^{\infty}\sqrt{n}e^{-i}\right)\\
&= O\left((w/r)^2\log^2(w/r+1)\sum_{i=1}^{\infty} i\log^2(i)\cdot e^{-i} + \sqrt{n}\sum_{i=1}^{\infty} e^{-i}\right)\\
&= O\left((w/r)^2\log^2(1+w/r) + \sqrt{n}\right).
\end{aligned}$$

Above we silently used Corollary 8 for $X = |V_i|$ and $\alpha = 2$. Now let us prove items (b) and (c). Let us first argue that the term $\mathbb{E}[|V_i|\log|V_i|]$ is, by Corollary 8, asymptotically dominated by the bound $\mathbb{E}[|V_i|]\cdot O(f^{\mathrm{E}}(i))$ on $\mathbb{E}[|E_i|]$ from Lemma 9. This follows by Lemmas 6 and 9 – if r is sufficiently large. Thus by plugging that bound into (4) we get

$$\begin{aligned}
&O\left(\sum_{i=1}^{\infty} \mathbb{E}[T_d(i)]e^{-i}\right)\\
&= \sum_{i=1}^{\infty}(w/r)^d i\cdot \min\{nr^d, (w/r)^{d-1}i\}e^{-i} + \sum_{i=1}^{\infty} n^{1-1/d}e^{-i}\\
&= O\left(\min\left\{nw^d\sum_{i\geq 1} ie^{-i}, (w/r)^{2d-1}\sum_{i=1}^{\infty} i^2e^{-i}\right\} + n^{1-1/d}\sum_{i=1}^{\infty} e^{-i}\right)\\
&= O\left(\min\left\{nw^d, (w/r)^{2d-1}\right\} + n^{1-1/d}\right).
\end{aligned}$$

◀

▶ Remark 12. The described distance oracle can be very easily made dynamic with only polylogarithmic overhead. That is, we can support insertions and deletions of vertices of the weighted unit-disk graph G, in amortized $O(\mathrm{polylog}\, n)$ time. To this end we simply replace the simplex range query data structure of Chan [5] that we build in the preprocessing with that of Matousek [16] which allows for polylogarithmic amortized updates to the point set and has only polylogarithmically slower preprocessing and query times.

4 Channels

The remaining part of the paper is devoted to proving the very convenient bound on $\bar{P}(i)$ from Lemma 5.

We start by introducing a notion of a *channel*, which is a parameterized grid-like object whose goal is to "discretize" the space of possible shortest $s \to t$ paths in $\mathrm{BB}(i)$. The next step is to upper-bound the probability that we fail to find reasonably short $s \to t$ path in the channel. Afterwards, we are ready to give explicit formulas for $i_{\max}$ and $W_{\mathrm{ub}}(i)$ so that the asymptotic bounds (6) and (7), as well as the bound $\bar{P}(i) \le e^{-i}$ hold.

Roughly speaking, a channel is a subset of vertices V restricted to some subspace. We generalize the channels defined in [18, page 41] to d-dimensional space and arbitrary start/end vertices s and t.

Recall that $w = ||t - s||$ and $w > r$. Let $K \ge 1$ be the smallest integer such that $l = w/(4K+1) \le r/4$. We also have

$$l = \frac{w}{4(K-1)+1} \cdot \frac{4(K-1)+1}{4K+1} > r/4 \cdot \frac{4K-3}{4K+1} \ge r/20. \tag{10}$$

We are going to work in the coordinate system introduced in Section 3.2. Let us denote the first axis by x_0 and the remaining axes by $x_1, \ldots, x_{d-1}$.

▶ **Definition 13** (Box $R(z_0, z_1, \ldots, z_{d-1})$). *Let $h > 0$ be fixed. Let us cut the space using planes $x_0 = lz$ and $x_i = (1/2 + z)h$ for all integers z and $i = 1, \ldots, d-1$.*

For $z_0, z_1, \ldots, z_{d-1} \in \mathbb{Z}$, the box $R(z_0, z_1, \ldots, z_{d-1})$ *contains all points $(x_i)_{i=0}^{d-1}$ satisfying:*

- $lz_0 \le x_0 \le l(z_0 + 1)$,
- $(-1/2 + z_i)h \le x_i \le (1/2 + z_i)h$ *for all* $i = 1, \ldots, d-1$.

Each box, defined as above, has size $l \times h \times \cdots \times h$. Note that $s \in R(0, 0, \ldots, 0)$ and $t \in R(4K, 0, \ldots, 0)$. Now suppose we want to travel from the box containing s to the box containing t using *jumps*, defined below.

▶ **Definition 14** (Jumping between boxes). *We say that we can* jump *from box $R(z_0, z_1, \ldots, z_{d-1})$ to box $R(z'_0, z'_1, \ldots, z'_{d-1})$ iff*

- $z'_0 = z_0 + 2$,
- $|z'_i - z_i| = 1$ *for all* $i = 1, \ldots, d-1$.

Consider a jumping trip from $R(0, 0, \ldots, 0)$ to $R(4K, 0, \ldots, 0)$.

▶ **Observation 15** (Reachable boxes). *Let $B = R(z_0, z_1, \ldots, z_{d-1})$ be an arbitrary box. Suppose a sequence of jumps (as defined above) from $R(0, 0, \ldots, 0)$ to $R(4K, 0, \ldots, 0)$ goes through the box B. Then, the following conditions hold:*

- $z_0 = 2k$ *for some integer* k, $0 \le k \le 2K$,
- $|z_i| \le \min(k, 2K - k)$ *for all* $i = 1, \ldots, d-1$,
- $z_i \equiv k \pmod 2$.

Now we are ready to define the *channel* parameterized by h.

▶ **Definition 16** (Channel). *A* channel $ch(h)$ *is a subset of $[0, 1]^d$ defined as the union of all boxes B satisfying the conditions of Observation 15.*

In other words, a channel $\mathrm{ch}(h)$ consists of all boxes that can appear in a sequence of jumps from the box containing s to the box containing t. Boxes, jumps, and channels are depicted in Figure 1.

In the following, we say that a box B is *empty* if it does not contain any vertex of G.

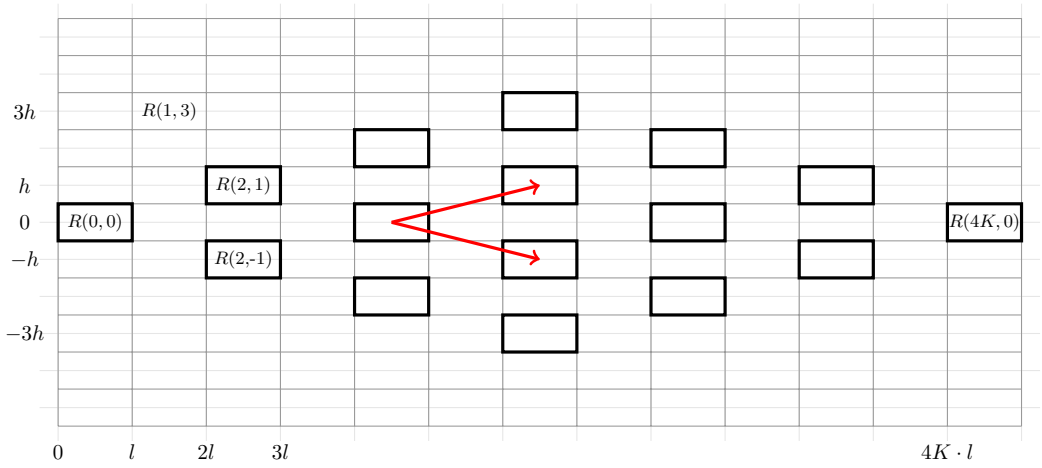


Figure 1 The rectangles represent boxes from Definition 13 for $d = 2$. The red arrows represent possible jumps from a single box. The channel $\mathrm{ch}(h)$ for $K = 3$ (see Definition 16) is represented by rectangles with thick black border.

4.1 Paths in a channel

Not all channels $\mathrm{ch}(h)$ are of our interest. We need a condition on h guaranteeing that if we can jump from a non-empty box B to another non-empty box B' then there exists an appropriate edge in the graph, namely if there is $u \in B \cap V$ and $v \in B' \cap V$ then $||u - v|| \leq r$. Then, a sequence of jumps between non-empty boxes will certify the existence of a path in G.

Observe that the distance between two opposite corners of B and B' (recall that B and B' have to satisfy Definition 14) is

$$\sqrt{(3l)^2 + (d-1)(2h)^2}.$$

We need this to be smaller than r. Taking into account that $l \leq r/4$, it is sufficient that

$$\left(\frac{3}{4}r\right)^2 + (d-1)(2h)^2 \leq r^2,$$

which gives

$$h \leq \frac{1}{8}\sqrt{\frac{7}{d-1}} \cdot r. \tag{11}$$

▶ **Definition 17** (Path in $\mathrm{ch}(h)$)**.** A path *in* $ch(h)$ *with* h *satisfying* (11) *is a sequence of non-empty boxes* $B_0, \ldots, B_{2K}$ *such that* $B_0 = R(0, 0, \ldots, 0)$, $B_{2K} = R(4K, 0, \ldots, 0)$, *and we can jump from* B_j *to* B_{j+1} *for all* $j = 0, \ldots, 2K - 1$.

Now we show that a path in $\mathrm{ch}(h)$ certifies the existence of an $s - t$ path in G which is not too long. Specifically, we show the following bound.

▶ **Lemma 18** (Channel induced path length)**.** *Suppose there is a path in* $ch(h)$*. Then, there exists an* $s - t$ *path in* G *of length no more than*

$$w\sqrt{1 + 40^2(d-1)(h/r)^2}. \tag{12}$$

Proof. Let $u_j = (u_0^j, \ldots, u_{d-1}^j)$ be a vertex of G in $B_j \cap V$. Additionally, set $u_0 = s$ and $u_{2K} = t$. Recall that u_j exists since each box in a path in $\mathrm{ch}(h)$ is non-empty. Consider subsequent vertices u_j and u_{j+1}. Note that

$$||u_{j+1} - u_j|| = \sqrt{\sum_{i=0}^{d-1}(u_i^{j+1} - u_i^j)^2} = (u_0^{j+1} - u_0^j)\sqrt{1 + \sum_{i=1}^{d-1}\left(\frac{u_i^{j+1} - u_i^j}{u_0^{j+1} - u_0^j}\right)^2}.$$

Recall that we have $u_0^{j+1} - u_0^j \geq l$ and $u_i^{j+1} - u_i^j \leq 2h$ for $i \geq 1$. Hence,

$$||u_{j+1} - u_j|| \leq (u_0^{j+1} - u_0^j)\sqrt{1 + (d-1)\frac{(2h)^2}{l^2}}.$$

Since $u_0 \to u_1 \to \ldots u_{2K}$ is a path in G, the length of a shortest $s - t$ path in G can be bounded by:

$$\begin{aligned}\sum_{j=0}^{2K-1} ||u_{j+1} - u_j|| &\leq \sqrt{1 + (d-1)\frac{(2h)^2}{l^2}} \cdot \sum_{j=0}^{2K-1}(u_0^{j+1} - u_0^j) \\ &= \sqrt{1 + (d-1)\left(\frac{2h}{l}\right)^2} \cdot w.\end{aligned}$$

The claimed bound is obtained by $l \geq r/20$. ◀

4.2 Probability

Denote by q the probability that a single box is empty. We have:

$$q = (1 - lh^{d-1})^n \leq \exp(-nlh^{d-1}). \tag{13}$$

Denote by $\hat{P}(h)$ the probability that no path exists in $\mathrm{ch}(h)$. We are going to prove the following lemma.

▶ **Lemma 19.** *There exists constants $q_0 \in (0,1)$ and $c > 0$ such that if $q < q_0$ then we have*

$$\hat{P}(h) \leq (cq)^{2^{d-3}}. \tag{14}$$

Proof. The proof will proceed by induction on d. We will thus use the notation $\hat{P}_d(h)$ and $\mathrm{ch}_d(h)$ to underline which dimension d we are currently referring to.

The crux of the proof is to prove the induction base $d = 2$, i.e., the bound

$$\hat{P}_2(h) \leq \sqrt{cq}$$

that holds for all $q < q_0$ for some constants c, q_0. Due to space constraints, the proof of this bound can be found in the full version. The general idea behind that proof is to reformulate it in terms of directed reachability in $n \times n$ grids: we want to bound the probability that no path between the corners of the grid exist when each vertex of the grid can fail with probability q. Then next step is to adapt the so-called *contour argument* from oriented percolation theory (see e.g., [10]) to work with finite grids.

For larger d it is enough to prove that the bound

$$\hat{P}_d(h) \leq \left(\hat{P}_{d-1}(h)\right)^2.$$

holds. Let $s \in \{-1, 1\}$. Consider a subchannel $\mathrm{ch}_d^s(h)$ of the channel $\mathrm{ch}_d(h)$ that is composed of the reachable boxes $B = R(z_0, z_1, \ldots, z_{d-1})$ fulfilling the following conditions:

- $z_0 = 2k$ for some integer k, $0 \le k \le 2K$,
- $|z_i| \le \min(k, 2K - k)$ for all $i = 1, \ldots, d-2$,
- $z_{d-1} = s \cdot \min(k, 2K - k)$,
- $z_i \equiv k \pmod 2$.

Observe that the above conditions say that B is a reachable box in $\mathrm{ch}_d(h)$ with additional constraint $z_{d-1} = s \cdot \min(k, 2K-k)$, which can also be written as $z_{d-1} = s \cdot \min(z_0, 4K - z_0)/2$.

Now one can see that $\mathrm{ch}_d^s(h)$ has exactly the same structure as $\mathrm{ch}_{d-1}(h)$: we can jump between boxes $R(z_0, \ldots, z_{d-2})$ and $R(z'_0, \ldots, z'_{d-2})$ in channel $\mathrm{ch}_{d-1}(h)$ if and only if we can jump between boxes

$$R(z_0, \ldots, z_{d-2}, s \cdot \min(z_0, 4K - z_0)/2)$$

and

$$R(z'_0, \ldots, z'_{d-2}, s \cdot \min(z'_0, 4K - z'_0)/2)$$

in channel $\mathrm{ch}_d^s(h)$. Therefore the probability that no path exists in $\mathrm{ch}_d^s(h)$ is bounded by $\hat{P}_{d-1}(h)$.

Observe that $\mathrm{ch}_d^{-1}(h)$ and $\mathrm{ch}_d^1(h)$ share only the corner boxes $R(0, 0, \ldots, 0)$ and $R(4K, 0, \ldots, 0)$. Thus if no path exists in $\mathrm{ch}_d(h)$, there must be no paths in $\mathrm{ch}_d^{-1}(h)$ and $\mathrm{ch}_d^1(h)$ independently. This clearly happens with probability at most $\left(\hat{P}_{d-1}(h)\right)^2$. ◀

5 Choosing the size of i-th bounding box

In this section we show how we derive the bound of Lemma 5 from Lemma 19. We will also be able to explicitly define the value $i_{\max}$ and the function $W_{\mathrm{ub}}(i)$ so that the asymptotic bounds (6) and (7) hold.

Suppose that for a fixed i we pick such h_i that $W_{\mathrm{ub}}(i) = w\sqrt{1 + 40^2(d-1)(h_i/r)^2}$. Then, by Lemma 18, a path in $\mathrm{ch}(h_i)$ certifies the existence of a $s \to t$ path in G of length at most $W_{\mathrm{ub}}(i)$. Such a path is clearly contained in $\mathrm{BE}(i)$, and thus also in $\mathrm{BB}(i)$. As a result, we conclude

$$\bar{P}(i) \le \hat{P}(h_i).$$

Given this, and since we want the probability $\bar{P}(i)$ to decay exponentially with i, we would like to choose h_i in a such way that $\hat{P}(h_i) \le e^{-i}$, which will imply $\bar{P}(i) \le e^{-i}$.

Suppose $\exp(-nlh^{d-1}) < q_0$, where q_0 is the constant of Lemma 19. By combining inequality (13) and the bound of Lemma 19, we have

$$\hat{P}(h) \le \exp\left(2^{d-3}(\log c - nlh^{d-1})\right).$$

In order to guarantee $\hat{P}(h_i) \le e^{-i}$, it is thus enough to have

$$2^{d-3}(\log c - nlh_i^{d-1}) \le -i$$
$$\log c + \frac{i}{2^{d-3}} \le nlh_i^{d-1}, \tag{15}$$

and

$$\log \frac{2}{q_0} \le nlh_i^{d-1}.$$

Let c' be such a positive constant that for $i \geq 1$ we have

$$\max\left(\log\frac{2}{q_0}, \log c + \frac{i}{2^{d-3}}\right) \leq c' \cdot i. \tag{16}$$

Now let h_0 be such that $h_0^{d-1} = \frac{c'}{nl}$, and let

$$h_i = h_0 \cdot i^{\frac{1}{d-1}}. \tag{17}$$

Then we have

$$\max\left(\log\frac{2}{q_0}, \log c + \frac{i}{2^{d-3}}\right) \leq c' \cdot i = c' \cdot \left(\frac{h_i}{h_0}\right)^{d-1} = c' \cdot h_i^{d-1} \cdot \frac{nl}{c'} = nlh_i^{d-1}.$$

So indeed, if h_i is defined as in (17), we have $\hat{P}(h_i) \leq e^{-i}$. So the explicit formula for $W_{\mathrm{ub}}(i)$ is:

$$W_{\mathrm{ub}}(i) = w\sqrt{1 + 40^2(d-1)\left(\frac{c'i}{nlr^{d-1}}\right)^{\frac{2}{d-1}}},$$

where c' is a constant defined in (16) and $l = \Theta(r)$ is as defined in (10). It is now verified that $W_{\mathrm{ub}}(i)$ indeed satisfies the asymptotic formula (7) from Section 3.

The above proof derivation of $\bar{P}(i) \leq e^{-i}$ is only correct if h_i is not too large. Namely, recall that the bound (11) requires that

$$h_i \leq \frac{1}{8}\sqrt{\frac{7}{d-1}} \cdot r. \tag{18}$$

Since h_i is an increasing function of i, this imposes a constraint on maximum possible $i = i_{\max}$ allowed. Hence, we need to have

$$\left(\frac{c' \cdot i_{\max}}{nl}\right)^{\frac{1}{d-1}} \leq \frac{1}{8}\sqrt{\frac{7}{d-1}} \cdot r.$$

$$i_{\max} = \left\lfloor \frac{1}{c'} \cdot \left(\frac{1}{8}\sqrt{\frac{7}{d-1}} \cdot r\right)^{d-1} \cdot nl \right\rfloor = \Theta(nr^d).$$

Observe that the above definition of $i_{\max}$ agrees with the bound (6) from Section 3.

References

1 Ittai Abraham, Daniel Delling, Amos Fiat, Andrew V. Goldberg, and Renato F. Werneck. Highway dimension and provably efficient shortest path algorithms. *J. ACM*, 63(5), 2016. doi:10.1145/2985473.

2 Ittai Abraham and Cyril Gavoille. Object location using path separators. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Principles of Distributed Computing*, PODC '06, page 188–197, New York, NY, USA, 2006. Association for Computing Machinery. doi:10.1145/1146381.1146411.

3 Hannah Bast, Daniel Delling, Andrew Goldberg, Matthias Müller-Hannemann, Thomas Pajor, Peter Sanders, Dorothea Wagner, and Renato F. Werneck. *Route Planning in Transportation Networks*, pages 19–80. Springer International Publishing, Cham, 2016. doi:10.1007/978-3-319-49487-6_2.

4 Sergio Cabello and Miha Jejcic. Shortest paths in intersection graphs of unit disks. *Comput. Geom.*, 48(4):360–367, 2015. doi:10.1016/j.comgeo.2014.12.003.

5 Timothy M. Chan. Optimal partition trees. *Discret. Comput. Geom.*, 47(4):661–690, 2012. doi:10.1007/s00454-012-9410-z.

6 Timothy M. Chan and Dimitrios Skrepetos. All-pairs shortest paths in unit-disk graphs in slightly subquadratic time. In Seok-Hee Hong, editor, *27th International Symposium on Algorithms and Computation, ISAAC 2016, December 12-14, 2016, Sydney, Australia*, volume 64 of *LIPIcs*, pages 24:1–24:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.ISAAC.2016.24.

7 Timothy M. Chan and Dimitrios Skrepetos. Approximate shortest paths and distance oracles in weighted unit-disk graphs. *J. Comput. Geom.*, 10(2):3–20, 2019. doi:10.20382/jocg.v10i2a2.

8 Panagiotis Charalampopoulos, Paweł Gawrychowski, Shay Mozes, and Oren Weimann. Almost optimal distance oracles for planar graphs. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2019, page 138–151, New York, NY, USA, 2019. Association for Computing Machinery. doi:10.1145/3313276.3316316.

9 Martin Dietzfelbinger and Friedhelm Meyer auf der Heide. A new universal class of hash functions and dynamic hashing in real time. In Mike Paterson, editor, *Automata, Languages and Programming, 17th International Colloquium, ICALP90, Warwick University, England, UK, July 16-20, 1990, Proceedings*, volume 443 of *Lecture Notes in Computer Science*, pages 6–19. Springer, 1990. doi:10.1007/BFb0032018.

10 Richard Durrett. Oriented percolation in two dimensions. *The Annals of Probability*, 12(4):999–1040, 1984. URL: http://www.jstor.org/stable/2243349.

11 P. Gupta and P. R. Kumar. Critical power for asymptotic connectivity. In *Proceedings of the 37th IEEE Conference on Decision and Control (Cat. No.98CH36171)*, volume 1, pages 1106–1110 vol.1, 1998. doi:10.1109/CDC.1998.760846.

12 P. Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, 46(2):388–404, 2000. doi:10.1109/18.825799.

13 Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Sci. Cybern.*, 4(2):100–107, 1968. doi:10.1109/TSSC.1968.300136.

14 Haim Kaplan, Wolfgang Mulzer, Liam Roditty, Paul Seiferth, and Micha Sharir. Dynamic planar voronoi diagrams for general distance functions and their algorithmic applications. *Discret. Comput. Geom.*, 64(3):838–904, 2020. doi:10.1007/s00454-020-00243-7.

15 Ken-Ichi Kawarabayashi, Philip N. Klein, and Christian Sommer. Linear-space approximate distance oracles for planar, bounded-genus and minor-free graphs. In *Proceedings of the 38th International Colloquim Conference on Automata, Languages and Programming - Volume Part I*, ICALP'11, page 135–146, Berlin, Heidelberg, 2011. Springer-Verlag.

16 Jirí Matousek. Efficient partition trees. *Discret. Comput. Geom.*, 8:315–334, 1992. doi:10.1007/BF02293051.

17 Mathew D. Penrose. On k-connectivity for a geometric random graph. *Random Structures & Algorithms*, 15(2):145–164, 1999.

18 Robert Sedgewick and Jeffrey Scott Vitter. Shortest paths in euclidean graphs. *Algorithmica*, 1(1):31–48, 1986. doi:10.1007/BF01840435.

19 Mikkel Thorup and Uri Zwick. Approximate distance oracles. *J. ACM*, 52(1):1–24, 2005. doi:10.1145/1044731.1044732.

20 Haitao Wang and Jie Xue. Near-optimal algorithms for shortest paths in weighted unit-disk graphs. *Discret. Comput. Geom.*, 64(4):1141–1166, 2020. doi:10.1007/s00454-020-00219-7.

No Krasnoselskii Number for General Sets

Chaya Keller
Department of Computer Science, Ariel University, Israel

Micha A. Perles
Einstein Institute of Mathematics, Hebrew University, Jerusalem, Israel

Abstract

For a family $\mathcal{F}$ of non-empty sets in $\mathbb{R}^d$, the Krasnoselskii number of $\mathcal{F}$ is the smallest m such that for any $S \in \mathcal{F}$, if every m or fewer points of S are visible from a common point in S, then any finite subset of S is visible from a single point. More than 35 years ago, Peterson asked whether there exists a Krasnoselskii number for general sets in $\mathbb{R}^d$. The best known positive result is Krasnoselskii number 3 for closed sets in the plane, and the best known negative result is that if a Krasnoselskii number for general sets in $\mathbb{R}^d$ exists, it cannot be smaller than $(d+1)^2$.

In this paper we answer Peterson's question in the negative by showing that there is no Krasnoselskii number for the family of all sets in $\mathbb{R}^2$. The proof is non-constructive, and uses transfinite induction and the well-ordering theorem.

In addition, we consider Krasnoselskii numbers with respect to visibility through polygonal paths of length $\leq n$, for which an analogue of Krasnoselskii's theorem for compact simply connected sets was proved by Magazanik and Perles. We show, by an explicit construction, that for any $n \geq 2$, there is no Krasnoselskii number for the family of compact sets in $\mathbb{R}^2$ with respect to visibility through paths of length $\leq n$. (Here the counterexamples are finite unions of line segments.)

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases visibility, Helly-type theorems, Krasnoselskii's theorem, transfinite induction, well-ordering theorem

Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.47

Funding *Chaya Keller*: Research partially supported by Grant 1065/20 from the Israel Science Foundation. Part of this research was performed while the author was affiliated with the Hebrew University and was supported by the Arianne de Rothschild fellowship, by the Hoffman Leadership program of the Hebrew University, and by an Advancing Women in Science grant of the Israel Ministry of Science and Technology.

1 Introduction

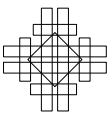
▶ **Definition 1.** *For a set $S \subset \mathbb{R}^d$ and two points $x, y \in S$, we say that "x sees y through S" or "y is visible from x through S" if S includes the segment $[x, y]$ or $x = y$. A set $S \subset \mathbb{R}^d$ is "starshaped" if some point $x \in S$ sees all other points in S. A set $S \subset \mathbb{R}^d$ is "finitely starlike" if every finite subset of S is visible from a common point.*

One of the best-known applications of Helly's theorem (that is, actually, equivalent to Helly's theorem, see [1]) is the following theorem, due to Krasnoselskii [12]:

▶ **Theorem 2** (Krasnoselskii)**.** *Let S be a non-empty compact set in $\mathbb{R}^d$. If every $d+1$ or fewer points of S are visible from a common point, then S is starshaped.*

In particular, the case $d = 2$ of the theorem asserts that if an art gallery is so shaped that for every three paintings there is a place where you can stand and see those three, then there is a place where you can stand and see all the paintings.

Krasnoselskii's theorem fails for unbounded closed sets. However, all known counterexamples satisfy the weaker requirement of being finitely starlike. This led Peterson [15] to ask the following:



▶ **Question 3** (Peterson). *Is there a Krasnoselskii number $K(d)$ such that for any infinite set $S \subset \mathbb{R}^d$, if every $K(d)$ points of S are visible from a single point, then S is finitely starlike?*

Peterson's question appears in the Handbook of Combinatorics [9, Chapter 14, Section 6.1.2, written by Erdős and Purdy] and in a treatise on open problems in geometry by Croft, Falconer and Guy [8, Chapter E1]. It was studied in a number of works. A few positive results were obtained, for special cases of sets that satisfy additional topological conditions [2, 4, 5]. Those include Krasnoselskii number 3 for closed sets in the plane, and Krasnoselskii numbers 4 and 5 under more complex conditions on the boundary of the set. On the other hand, the complement of the closed unit cube in $\mathbb{R}^d$ shows that if a Krasnoselskii number for open sets in $\mathbb{R}^d$ exists, it must be $\geq 2d$. Breen [4] showed by an explicit example that if a Krasnoselskii number for F_σ sets in the plane exists, then it must be ≥ 9. This result was generalized to any $d \geq 2$, with the lower bound $(d+1)^2$, by Shlezinger [16]. For other partial results and related problems, see [3] and the references therein.

In this paper we answer Peterson's question in the negative by showing that the Krasnoselskii number $K(2)$ does not exist, and consequently, $K(d)$ does not exist for any $d > 2$, since a planar construction can be embedded in $\mathbb{R}^d$ for any $d > 2$. Formally, we prove the following:

▶ **Theorem 4.** *For any $k \geq 2$, there exists a set $S \subset \mathbb{R}^2$ such that any $2k+3$ points in S are visible from a common point, but some $2k+4$ points in S are not simultaneously seen from any point.*

Unlike Breen's construction that leads to an F_σ-set, our proof is non-constructive and uses transfinite induction and the well-ordering theorem. We note that we do use the axiom of choice which is equivalent to the well-ordering theorem, but we do not use any additional hypothesis except for the standard ZFC system of axioms.

In the second part of the paper, we consider a generalization of the notion of visibility, namely visibility through polygonal paths.

▶ **Definition 5.** *For a set $S \subset \mathbb{R}^d$ and $x, y \in S$, we say that "x sees y through S by a polygonal path of length n" or "y is visible from x through S by a polygonal path of length n" if there exist points $x_1, x_2, \ldots, x_{n-1}$ and a polygonal path*

$$\langle x, x_1, \ldots, x_{n-1}, y\rangle = [x, x_1] \cup [x_1, x_2] \cup \ldots \cup [x_{n-1}, y]$$

that lies entirely within S.

This definition of visibility is closely related to the notion of L_n-sets. These are sets that are connected through polygonal paths of length $\leq n$. This notion was defined by Horn and Valentine [10], and studied in numerous papers (see, e.g., [7, 14, 17]). Of course, the original notion of visibility considered above corresponds to the case of visibility through polygonal paths of length $n = 1$.

In [14, Theorem 1.2], Magazanik and Perles proved a Krasnoselskii-type theorem with respect to visibility through polygonal paths. Formally, they showed that for any n and for any compact and simply connected set $S \subset \mathbb{R}^2$, if every three points in S are visible from a common point through polygonal paths of length $\leq n$, then all points of S are visible from a single point through polygonal paths of length $\leq n$.

This raises the natural analogue of Peterson's question for this notion of visibility: Is there a number $K'(2, n)$ such that for any set $S \subset \mathbb{R}^2$, if every $K'(2, n)$ or fewer points of S are visible from a common point through polygonal paths of length $\leq n$, then every finite subset of S is visible from a common point through polygonal paths of length $\leq n$?

We answer this question in the negative. Namely, we prove, by constructing a sequence of explicit examples, the following:

▶ **Theorem 6.** *For any $n, k \geq 2$, there exists a compact set $S \subset \mathbb{R}^2$ such that every k points in S are visible from a common point through polygonal paths of length $\leq n$, but some $k+1$ points of S are not visible from a common point through polygonal paths of length $\leq n$.*

The sets S we construct in the proof of Theorem 6 are actually finite unions of closed line segments.

A similar result with respect to the related concept of visibility through *staircase* paths of length $\leq n$ was obtained by Breen [6]. However, the construction of [6] is significantly different from ours.

This paper is organized as follows: In Section 2 we study Peterson's problem with respect to the classical definition of visibility and prove Theorem 4. In Section 3 we study visibility through polygonal paths of length $\leq n$ and prove Theorem 6.

2 Proof of Theorem 4

2.1 Some basic facts from set theory

In this subsection we recall a few basic definitions and facts from set theory that will be used in our proof. For the sake of brevity, we present the definitions in a somewhat informal way. A formal treatment can be found in standard textbooks on basic set theory, e.g., [11, 13].

Well ordering. Let S be a set. A binary relation $<$ on S is called a *well-ordering* if:

1. For any $x, y \in S$, exactly one of the following holds: either $x < y$, or $y < x$, or $x = y$.
2. " $<$ " is transitive.
3. Any non-empty subset S' of S has a first element x_0, namely, $\forall \emptyset \neq S' \subset S, \exists x_0 \in S' : \forall x \in S', x_0 \leq x$.

A pair $(S, <)$ where $<$ is a well-ordering on S is called a *well-ordered set*.

Transitive sets and ordinals. A set S is called *transitive* if any element of S is a subset of S, that is, if $(x \in S) \wedge (y \in x) \Rightarrow (y \in S)$. A set O is called an *ordinal* if it is transitive and $(O, \in)$ (i.e., O with the relation $(x < y) \Leftrightarrow (x \in y)$) is a well-ordered set.

Initial section. Let $(S, <)$ be a well-ordered set. An initial section of S is a subset $S' \subset S$ of the form $S' = \{x \in S : x < y\}$, for some $y \in S$.

Isomorphism. Two well-ordered sets $(S, <_S)$ and $(T, <_T)$ are *isomorphic* if there exists a bijection $f : S \to T$ such that $\forall x, y \in S : (x <_S y) \Leftrightarrow (f(x) <_T f(y))$.

Two basic facts on ordinals.

1. For any set $\mathcal{O}$ of ordinals, $(\mathcal{O}, \in)$ is a well-ordered set. In particular, any set of ordinals contains a minimal element with respect to the relation $\in$.
2. Any well-ordered set $(S, <)$ is isomorphic to a unique ordinal.

The well-ordering theorem. For any set S, there exists a relation $<$ on S such that $(S, <)$ is a well-ordered set. (This theorem may be considered an axiom, being equivalent to the Axiom of Choice).

Initial ordinal. For a set S, consider all possible well-orderings of S. Each of them is isomorphic to a unique ordinal. The corresponding set of ordinals has a minimal element (with respect to the relation $\in$). This element is called *the initial ordinal* that corresponds to S.

Cardinals. Informally, the cardinality of a set S measures its "size". Formally, the cardinal of S, denoted by $|S|$, is identified with the initial ordinal that corresponds to S. This allows comparing the "sizes" of any two sets, as any two ordinals are comparable by the relation $\in$. It is clear that we have $|S| \leq |T|$ if and only if there exists an injection $f : S \to T$.

Arithmetic of infinite cardinals. We use two basic facts from cardinal arithmetic. If S, T are infinite sets then $|S \times T| = |S \cup T| = \max(|S|, |T|)$. If, in addition, $|S| < |T|$, then $|T \setminus S| = |T|$.

Initial section of an initial ordinal. It follows from the above definitions that if O is the initial ordinal that corresponds to some set S and O' is an initial section of O, then $|O'| < |O|$.

Induction on ordinals and transfinite induction. The induction principle on ordinals implies the following: Let O be an ordinal, and let P be a property of ordinals. If for any $O' \in O$, we have $(\forall O'' < O' : P(O'')) \Rightarrow P(O')$, then $\forall O' \in O, P(O')$. As any well-ordered set corresponds to a unique ordinal, this method can be used to prove claims on general well-ordered sets. Its application is commonly called *transfinite induction*.

Application in our case. A classical way to use transfinite induction to prove an assertion on a general set S is to use the well-ordering theorem to define a well-ordering $<$ such that $(S, <)$ is isomorphic to the initial ordinal that corresponds to S, and then to use transfinite induction to prove the assertion for $(S, <)$. An important feature deployed in this method is that for any initial section S' of $(S, <)$, we have $|S'| < |S|$, as was explained above.

We shall apply this method to the set S all k-tuples of points in the lower half-plane (whose cardinality is clearly $\mathfrak{c} = 2^{\aleph_0}$), and the fact that the cardinality of any initial section in our ordering is strictly smaller than $\mathfrak{c}$ will play a crucial role in our proof.

2.2 Proof of the theorem

In this subsection we prove Theorem 4. Let us recall its statement.

Theorem 4. For any $2 \leq k \in \mathbb{N}$, there exists a set $T = T(k) \subset \mathbb{R}^2$ such that the following holds:

1. Any $2k + 3$ points of T are seen, through T, from a common point $x \in T$.
2. There are $2k + 4$ points in T that are not seen, through T, from any point of T.

We divide the presentation into three parts. First, we reduce the problem to proving the existence of a subset S of the real line that satisfies certain conditions, called a *k-shutter*. Then, we describe the transfinite induction argument, without getting into the geometric details. Finally, we present a formal proof that fills in the geometric part.

2.2.1 Reduction to proving the existence of a k-shutter

Let us denote by $\mathbb{R}^2_+$ the upper open half-plane $\mathbb{R}^2_+ = \{(x, y) \in \mathbb{R}^2 | y > 0\}$, and by $\mathbb{R}^2_-$ the lower open half-plane $\mathbb{R}^2_- = \{(x, y) \in \mathbb{R}^2 | y < 0\}$. For two points $x \in \mathbb{R}^2_+, y \in \mathbb{R}^2_-$, and a subset S of the x-axis, $S \subset \{(x, 0) | x \in \mathbb{R}\}$, we say that x *sees* y *via* S, if x sees y through $\mathbb{R}^2_+ \cup S \cup \mathbb{R}^2_-$. (Note that this last definition is not the same as visibility *through* S defined above.)

The following definition plays a central role in the proof of Theorem 4.

▶ **Definition 7.** *A subset S of the x-axis, $S \subset \{(x, 0) | x \in \mathbb{R}\}$, is called a* k-shutter*, if:*

(a) *For any k-tuple $\{a_1, \ldots, a_k\} \subset \mathbb{R}^2_-$ there exists a point $z \in \mathbb{R}^2_+$ that sees each a_i $(1 \leq i \leq k)$ via S, and*

(b) *There exists a $(k + 1)$-tuple $K = \{y_1, \ldots, y_{k+1}\} \subset \mathbb{R}^2_-$ such that no point in $\mathbb{R}^2_+$ sees all the points of K via S.*

Note that, by symmetry, the definition of a k-shutter is not sensitive to interchanging $\mathbb{R}^2_+$ and $\mathbb{R}^2_-$. In addition, it is easy to see that for $k \geq 2$, any k-shutter S satisfies $|S| \geq 2$ and does not include a segment.

▶ **Observation 8.** *In order to prove Theorem 4, it suffices to prove that for any $k \in \mathbb{N}$, there exists a k-shutter S.*

Proof of Observation 8. Given a k-shutter S, let $T = \mathbb{R}^2_+ \cup \mathbb{R}^2_- \cup S$, and let T' be a set of $2k+3$ points of T. If $T' \cap S = \emptyset$, then any point of S sees all points of T' through T. If $T' \cap S$ consists of a single point x, then x sees all the points of T' through T. If $|T' \cap S| \geq 2$, then by the pigeonhole principle, either $T' \cap \mathbb{R}^2_-$ or $T' \cap \mathbb{R}^2_+$ contains at most k points. W.l.o.g., assume $|T' \cap \mathbb{R}^2_-| \leq k$. As S is a k-shutter, some point $z \in \mathbb{R}^2_+$ sees $T' \cap \mathbb{R}^2_-$ through T. The point z clearly sees all points of T' through T.

On the other hand, take $K_1 \subset \mathbb{R}^2_-$ to be a $(k+1)$-set that is not seen via S by any point in $\mathbb{R}^2_+$, and take $K_2 \subset \mathbb{R}^2_+$ to be a $(k+1)$-set that is not seen via S by any point in $\mathbb{R}^2_-$. Since $|S| \geq 2$ and S does not include a segment, there exist two points $s_1, s_2 \in S$ that do not see each other through S. Then the set $K_1 \cup K_2 \cup \{s_1, s_2\} \subset T$ is a $(2k+4)$-sized subset of T that is not seen by any point through T. ◀

2.2.2 The transfinite induction argument

To prove the existence of a k-shutter, we use transfinite induction. Let $K = \{y_1, \ldots, y_{k+1}\} \subset \mathbb{R}^2_-$ be a fixed $(k+1)$-set of points in $\mathbb{R}^2_-$, and let $U = \{\{a_1, a_2, \ldots, a_k\} \subset \mathbb{R}^2_-\}$ be the family of all k-subsets of $\mathbb{R}^2_-$. Clearly, $|U| = \mathfrak{c}$. Let O be the initial ordinal of cardinality $\mathfrak{c}$, namely, $|O| = \mathfrak{c}$ and for each $\lambda \in O$, $|\{\alpha \in O : \alpha < \lambda\}| < \mathfrak{c}$. Let $\lambda \mapsto u_\lambda$ $(\lambda \in O)$ be a bijection between O and U (whose existence follows from the well-ordering theorem). We shall prove the existence of a k-shutter S such that each k-set in U is seen via S by some $z \in \mathbb{R}^2_+$, while no $z \in \mathbb{R}^2_+$ sees the $(k+1)$-set K via S.

We construct S by a transfinite induction process, that corresponds to induction on the ordinal O. We index the steps of the process by the elements of O, and in step λ we "take care" of the k-set $a^\lambda = \{a^\lambda_1, \ldots, a^\lambda_k\}$ that corresponds (in the isomorphism) to $\lambda \in O$.

During the process, we construct two increasing sequences of subsets of the x-axis: $\{A_\lambda\}_{\lambda \in O}$ and $\{B_\lambda\}_{\lambda \in O}$ (where "increasing" means that if $\lambda_1 < \lambda_2$ then $A_{\lambda_1} \subseteq A_{\lambda_2}$ and $B_{\lambda_1} \subseteq B_{\lambda_2}$). The sets A_λ contain points that will be included in S, while the sets B_λ contain points that will not be included in S. Of course, we make sure that the B_λ's are disjoint from the A_λ's. In addition, we make sure that at each step λ, we have $|A_\lambda|, |B_\lambda| \leq \max(|\lambda|, \aleph_0)$.

At Step λ, we define A_λ and B_λ by adding points to the sets $A^0_\lambda = \bigcup_{\lambda' < \lambda} A_{\lambda'}$ and $B^0_\lambda = \bigcup_{\lambda' < \lambda} B_{\lambda'}$, respectively. First we add at most $\max(|\lambda|, \aleph_0)$ points to B^0_λ to form B_λ in a certain way to be explained shortly. Then we add $k-1$ (or fewer) points to A^0_λ to form A_λ, in such a way that the set $a^\lambda = \{a^\lambda_1, \ldots, a^\lambda_k\}$ is seen through A_λ by some $z \in \mathbb{R}^2_+$, while the set K is not seen through A_λ by any $z \in \mathbb{R}^2_+$. The points added to B_λ are responsible for the latter condition – they are chosen in such a way that forcing A_λ to avoid them will guarantee that K is not seen via A_λ from any point in the upper open half-plane.

At the end of the process, we set $S = \bigcup_{\lambda \in O} A_\lambda$. It is clear from the construction that any k-set $a^\lambda \in U$ is seen via S by some $z \in \mathbb{R}^2_+$ (since it is seen via A_λ) and also that K is not seen via S by any $z \in \mathbb{R}^2_+$ (as otherwise, assuming that for any $1 \leq i \leq k+1$, y_i is seen via some A_{λ_i}, K would be seen via A_λ for $\lambda = \max\{\lambda_1, \ldots, \lambda_{k+1}\}$, contradicting the construction).

The crucial observation that makes the definition of the sets A_λ and B_λ possible is that the cardinality of any initial section of O is smaller than $\mathfrak{c}$, and thus, at any step λ of the process we have $|A^0_\lambda|, |B^0_\lambda| < \mathfrak{c}$. This means that the sets A^0_λ and B^0_λ cover only a "very small" part of the x-axis, and so we have "enough room" to select the points we need to add in order to define B_λ and A_λ, as will be shown formally below.

2.2.3 Formal proof

We prove the following proposition.

▶ **Proposition 9.** *There exist increasing sequences of sets $\{A_\lambda\}_{\lambda\in O}$ and $\{B_\lambda\}_{\lambda\in O}$, such that, for any $\lambda \in O$, we have:*

1. $A_\lambda, B_\lambda \subset \{(x,0)|x \in \mathbb{R}\}$;
2. $A_\lambda \cap B_\lambda = \emptyset$;
3. $|A_\lambda|, |B_\lambda| \leq \max(|\lambda|, \aleph_0) < \mathfrak{c}$;
4. *The k-set $a^\lambda = \{a^\lambda_1, \ldots, a^\lambda_k\}$ is seen via A_λ by some $z \in \mathbb{R}^2_+$, while the $(k+1)$-set $K = (y_1, \ldots, y_{k+1})$ is not seen via A_λ by any $z \in \mathbb{R}^2_+$.*
5. $\forall 1 \leq i < j \leq k+1$, *$\ell(y_i, y_j) \cap \{(x,0)|x \in \mathbb{R}\} \subset B_\lambda$, where $\ell(y_i, y_j)$ is the line spanned by y_i and y_j.*
6. *For any $z \in \mathbb{R}^2_+$ such that z sees at least two points of K via $\bigcup_{\lambda'<\lambda} A_{\lambda'}$, there exists $1 \leq i \leq k+1$ such that $\{(x,0)|x \in \mathbb{R}\} \cap [z, y_i] \in B_\lambda$.*

Proposition 9 implies Theorem 4, since the set $S = \bigcup_{\lambda\in O} A_\lambda$ is a k-shutter, as was explained in Section 2.2.2. Note that the inductive step is performed for any $\lambda \in O$, namely, for any $\lambda < \mathfrak{c}$ (and not for $\lambda = O$), and the set S is a union of $\mathfrak{c}$ sets.

Proof. The proof is by transfinite induction on the elements of O.

Induction basis. We start by defining A_0 and B_0. (Note that there is no formal need to present the initial step separately. However, in our case this step is somewhat different from the other ones, and thus we have to present it.)

Consider all lines that are spanned by two points $y_i, y_j \in K$. Let B_0 be the set of all intersection points of these lines with the x-axis. Clearly, $|B_0| \leq \binom{k+1}{2}$, and B_0 satisfies condition 5. Condition 6 is satisfied vacuously, since $\bigcup_{\lambda'<0} A_{\lambda'} = \emptyset$.

In order to define A_0, consider all lines that pass through at least one point of $a^0 = \{a^0_1, \ldots, a^0_k\}$ and at least one point of B_0. The union of these lines does not cover the upper half-plane (being a finite union of lines), hence, there exists a point $z \in \mathbb{R}^2_+$ that is not incident with any such line. Denote the intersections of the segments $[z, a^0_1], \ldots, [z, a^0_k]$ with the x-axis by $s_1, \ldots, s_t$ for some $t \leq k$, and set $A_0 = \{s_1, \ldots, s_t\}$. Note that by the construction of B_0, $A_0 \cap B_0 = \emptyset$.

Both A_0 and B_0 are finite, and so we have $|A_0|, |B_0| \leq \max(0, \aleph_0)$. The point z sees all k points in $a^0 = \{a^0_1, \ldots a^0_k\}$ via A_0. Furthermore, no point in $\mathbb{R}^2_+$ can see K via A_0. Indeed, $|A_0| \leq k$, and thus, by the pigeonhole principle, if some $z \in \mathbb{R}^2_+$ sees K via A_0, it must see two points y_i, y_j via the same point $s_\ell \in A_0$. In such a case, $s_\ell \in B_0$ by the definition of B_0, contradicting the construction which assures that $A_0 \cap B_0 = \emptyset$. Therefore, A_0 and B_0 satisfy the assertion of the proposition.

Induction step. For $\lambda \in O$, we describe the definition of A_λ and B_λ, assuming that for any $\lambda' < \lambda$, $A_{\lambda'}$ and $B_{\lambda'}$ have been defined and satisfy the assertion. As described above, we denote

$$A^0_\lambda = \cup_{\lambda'<\lambda} A_{\lambda'} \qquad \text{and} \qquad B^0_\lambda = \cup_{\lambda'<\lambda} B_{\lambda'}.$$

(Note that if $\lambda = \bar{\lambda} + 1$ is a successor ordinal, then we have $A^0_\lambda = A_{\bar{\lambda}}$ and $B^0_\lambda = B_{\bar{\lambda}}$.)

Defining B_λ. Consider the family $\bar{\mathcal{L}}$ of lines that pass through at least one point of A^0_λ and at least one of the points of $K = \{y_1, \ldots, y_{k+1}\}$. Let $Z \subset \mathbb{R}^2_+$ be the set of all intersection points of two lines in $\bar{\mathcal{L}}$ that lie in $\mathbb{R}^2_+$. Fix $z \in Z$. Since z does not see all $k+1$ points of K via A^0_λ (by the induction hypothesis), there exists $1 \leq i \leq k+1$ such that the intersection point between $[z, y_i]$ and the x-axis is not in A^0_λ. We add such an intersection point to B^0_λ (if there are several such points, we just add one of them arbitrarily). We do this process for all $z \in Z$, thus condition 6 is satisfied. The resulting set is B_λ.

Note that since $|A^0_\lambda|, |B^0_\lambda| \leq \max\{|\lambda|, \aleph_0\}$, and since $|\lambda|^2 \leq \max\{|\lambda|, \aleph_0\}$, it follows that $|B_\lambda| \leq \max\{|\lambda|, \aleph_0\}$. Furthermore, by construction, $B_\lambda \cap A^0_\lambda = \emptyset$ and $B_0 \subset B^0_\lambda \subset B_\lambda$, thus condition 5 is satisfied.

Motivation behind the definition of B_λ. In the definition of A_λ to be presented below, we add at most $k-1$ points to A^0_λ. As no point $z \in \mathbb{R}^2_+$ can see more than a single point of K via the same point in A_λ (due to the definition of B_0), a point $z \in \mathbb{R}^2_+$ can see K via A_λ only if it sees at least two points of K via A^0_λ. Z is the set of all "dangerous" points in the upper half-plane, that see via A^0_λ at least two points of K. By our definition of B_λ, we ensure that once we define A_λ such that $A_\lambda \cap B_\lambda = \emptyset$, no point in Z will see via A_λ all $k+1$ points of K. This guarantees that no point $z \in \mathbb{R}^2_+$ can see K via A_λ.

Defining A_λ. A_λ is obtained by adding at most $k-1$ points to A^0_λ. These points are chosen such that $a^\lambda = \{a^\lambda_1, \ldots, a^\lambda_k\}$ will be seen via A_λ by some point z in the upper open half-plane. Furthermore, the point z will be a point that sees a^λ_1 via A^0_λ. (This condition allows us to add only $k-1$ points, rather than k points.)

Let $x \in A^0_\lambda$ be chosen arbitrarily, and let ℓ be the line that passes through x and a^λ_1. Let

$$\mathcal{L} = \{\ell(a^\lambda_i, b) | 2 \leq i \leq k, b \in B_\lambda\}$$

be the set of all lines $\ell(a^\lambda_i, b)$ that pass through some point $a^\lambda_i \in a^\lambda$, $(2 \leq i \leq k)$, and some point $b \in B_\lambda$.

Since $|B_\lambda| \leq \max(|\lambda|, \aleph_0) < \mathfrak{c}$, and since any line in $\mathcal{L}$ intersects ℓ in at most one point in the upper open half-plane, it follows that there exists a point $z \in \ell \cap \mathbb{R}^2_+$ that is not covered by any line of $\mathcal{L}$. (Recall that no line in $\mathcal{L}$ coincides with ℓ, since ℓ intersects the x-axis in $x \in A^0_\lambda$, while any line in $\mathcal{L}$ intersects the x-axis in some point of B_λ.)

The set A_λ is obtained from A^0_λ by adding the intersections of the segments $[z, a^\lambda_i]$, for $2 \leq i \leq k$, with the x-axis. Note that the intersection of $[z, a^\lambda_1]$ with the x-axis is included in A^0_λ so there is no need to add it. Also note that it is possible that some of the $k-1$ "added" points already belong to A^0_λ, and so fewer than $k-1$ points are added.

Obviously, $A^0_\lambda \subset A_\lambda$ and $|A_\lambda| \leq \max(|\lambda|, \aleph_0)$. By the construction, the added points are not in B_λ, and hence, $A_\lambda \cap B_\lambda = \emptyset$. Furthermore, the point z sees $a^\lambda_1, \ldots, a^\lambda_k$ via A_λ. Finally, the definition of B_λ guarantees that no $z \in \mathbb{R}^2_+$ sees K via A_λ, thus condition 4 is also satisfied. (See the motivation above.) Therefore, A_λ and B_λ satisfy the assertion.

By transfinite induction, this completes the proof of the proposition, and thus, also of the theorem. ◀

▶ Remark 10. One can strengthen the assertion of Theorem 4 by considering a countable dense collection of "forbidden" $(k+1)$-sets, instead of a single $(k+1)$-set K. No significant change in the proof method is needed.

3 Proof of Theorem 6

In this section we prove that no Krasnoselskii-type theorem exists for the notion of visibility through a polygonal path of length $\leq n$ (in short, an *n-path*). By constructing a sequence of planar sets $S_{n,k}$, $n, k \geq 2$, each consisting of a finite union of closed segments, we prove:

Theorem 6 (Restatement). For any $n, k \geq 2$, there exists a set $S = S_{n,k} \subset \mathbb{R}^2$ such that every k points in S are visible from a common point through n-paths in S, but some $k+1$ points of S are not visible from a common point through n-paths in S.

Proof. The proof consists of two steps. In the first step we prove the existence of $S_{2,k}$ for any $k \geq 2$, and in the second step we extend the construction of the first step to obtain $S_{n,k}$ for $n > 2$.

Step 1: Constructing $S_{2,k}$, $k \geq 2$. We obtain $S_{2,k}$ by an appropriate planar embedding of the complete bipartite graph $K_{k+1,k+1}$ with one perfect matching removed. Let $P \subset \mathbb{R}^2$ be a convex $(2k+2)$-gon. We assume that no three diagonals of P have a common point inside P. (This can be achieved by slightly perturbing the vertices of P.)

Label the vertices of P cyclically (clockwise) $a_0, b_0, a_1, b_1, \ldots, a_k, b_k$. We regard the indices as numbers modulo $k+1$ (see Figure 1). Define $\kappa = \lfloor \frac{k}{2} \rfloor$.

The matching to be removed will be

$$\{[a_i, b_{i+\kappa}] : i = 0, 1, \ldots, k\} = \{[a_{i-\kappa}, b_i] : i = 0, 1, \ldots, k\}.$$

Note that the segments $[a_i, b_{i+\kappa}]$ are diagonals, not boundary edges of P.

Define, therefore, for $i = 0, 1, \ldots, k$,

$$B_i = \bigcup \{[a_\nu, b_i] : \nu = 0, 1, \ldots, k, \qquad \nu \neq i - \kappa\},$$

and let

$$S_{2,k} = \bigcup_{i=0}^{k} B_i.$$

$S_{2,2}$ is just a hexagon. $S_{2,3}$ is an octagon with four vertex-disjoint diagonals of order 3. See Figure 1 for $S_{2,4}$ and $S_{2,5}$.

Claims 11 and 12 below assert that in $S_{2,k}$, every k points are visible from a common point through a 2-path, but there exist $k+1$ points in $S_{2,k}$ that are not visible from any common point through 2-paths.

▷ Claim 11. For any k points $x_1, \ldots, x_k \in S_{2,k}$ there exists a point $z \in S_{2,k}$ that sees all of them through 2-paths in $S_{2,k}$.

Proof. Each point x_i $(1 \leq i \leq k)$ belongs to some B_j, $0 \leq j \leq k$. (If x_i lies in two B_j's, choose one of them arbitrarily.) Hence we may assume, without loss of generality, that $x_1, \ldots, x_k$ are all contained in $B_1 \cup \ldots \cup B_k$. It follows that $a_{k+1-\kappa}$ sees each x_i through a 2-path. Indeed, $a_{k+1-\kappa}$ is connected by a segment in $S_{2,k}$ to each of $b_1, \ldots, b_k$, and each b_i is connected by a segment in $S_{2,k}$ to every point in B_i. ◁

▷ Claim 12. For $i = 0, 1, \ldots, k$, let c_i be the midpoint of $[b_i, a_{i+1}]$, where the indices are taken modulo $k+1$. Then $c_0, \ldots, c_k$ are not visible from any common point in $S_{2,k}$ through 2-paths.

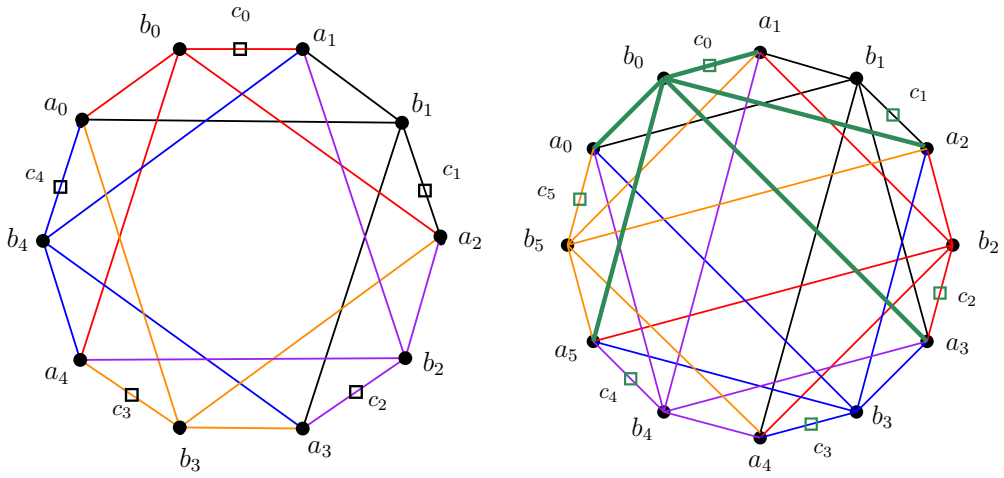


Figure 1 The left figure is $S_{2,4}$ and the right figure is $S_{2,5}$. Each B_i is colored in a different color.

Proof. The cases $k = 2, 3$ are easy to verify. Let $k \geq 4$ and assume to the contrary that some point $z \in S_{2,k}$ sees all the points $c_0, \ldots, c_k$ through 2-paths in $S_{2,k}$. It follows that for each $0 \leq i \leq k$, z sees b_i or a_{i+1} through a 1-path, and in particular, the set of vertices of P that z sees through a 1-path is of size $\geq k+1 \geq 5$. We reach a contradiction, by considering three cases:

Case 1: z is a vertex of P. Thus $z = a_i$ or $z = b_i$ for some i, $0 \leq i \leq k$. But a_i sees neither $b_{i+\kappa}$, nor $a_{i+\kappa+1}$ through a 1-path in $S_{2,k}$, and b_i sees neither $b_{i-\kappa-1}$, nor $a_{i-\kappa}$ through a 1-path in $S_{2,k}$.

Case 2: z is not a vertex of P, but z lies on a boundary edge of P. In this case, z sees through a 1-path only two vertices of P, a contradiction.

Case 3: z is not a vertex of P, but z lies on a diagonal of P. In this case, z lies on at most two diagonals of P. (Here we use the assumption that no three diagonals of P have a common point inside P.) Thus, z sees through a 1-path at most four vertices of P. This is a contradiction, as z must see at least 5 vertices of P, as was explained above.

This completes the proof. ◁

Step 2: Construction of $S_{n,k}$ for $n > 2, k \geq 2$. Given $n > 2$ and $k \geq 2$, we modify the construction of $S_{2,k}$ to obtain $S_{n,k}$, in the following way: We consider $S_{2,k}$, and for each $0 \leq i \leq k$ we add a simple polygonal path Γ_i of length $n-2$ (with no two consecutive edges collinear) that starts at c_i and lies outside the $(2k+2)$-gon P, such that the paths Γ_j $(0 \leq j \leq k)$ are pairwise disjoint (see Figure 2). For $0 \leq i \leq k$, define $C_i = B_i \cup \Gamma_i$, and let

$$S_{n,k} = \bigcup_{i=0}^{k} C_i.$$

▷ Claim 13. $S_{n,k}$ satisfies the assertion of Theorem 6.

Proof. Same as in the proof of Claim 11, given $x_1, \ldots, x_k \in S_{n,k}$ we may assume that $\{x_1, \ldots, x_k\} \subset C_1 \cup \ldots \cup C_k$. In this case, $a_{k+1-\kappa}$ sees each of the points $b_1, \ldots, b_k$ through a 1-path, and thus, sees every point in $C_1 \cup \ldots \cup C_k$ through an n-path.

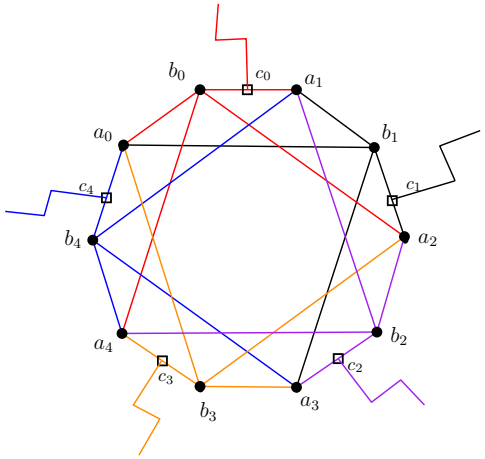


Figure 2 An illustration of $S_{5,4}$. Each C_i is colored in a different color.

On the other hand, we claim that the $k+1$ outer endpoints of $\Gamma_0, \ldots, \Gamma_k$ are not seen by any common point x through n-paths. Indeed, assume on the contrary that some $x \in S_{n,k}$ sees all these points through n-paths. If $x \in S_{2,k}$, then this assumption clearly implies that x sees all the points $c_0, \ldots, c_k$ through 2-paths, contradicting Claim 12. If $x \notin S_{2,k}$ then $x \in \Gamma_i$ for some i. In this case, our assumption implies that even c_i sees the $k+1$ outer endpoints of $\Gamma_0, \ldots, \Gamma_k$ through n-paths. However, $c_i \in S_{2,k}$, and thus we obtain a contradiction, same as in the first case. ◁

This completes the proof of Theorem 6. ◀

4 Open problems

The proof method of Theorem 4 is not constructive, and the resulting set T does not admit any "nice" topological structure (e.g., being a Borel set). Thus, it will be interesting to determine whether one may add some topological restriction on T to the conditions of Theorem 4.

References

1 J. Borwein. A proof of the equivalence of Helly's and Krasnosselsky's theorems. *Canad. Math. Bull.*, 20:35–37, 1977.
2 M. Breen. Clear visibility, starshaped sets, and finitely starlike sets. *J. Geometry*, 19:183–196, 1982.
3 M. Breen. Krasnoselskii-type theorems. *Ann. New York Acad. Sci.*, 440:142–146, 1985.
4 M. Breen. Some Krasnosel'skii numbers for finitely starlike sets in the plane. *J. Geometry*, 32:1–12, 1988.
5 M. Breen. Finitely starlike sets whose F-stars have positive measure. *J. Geometry*, 35:19–25, 1989.
6 M. Breen. A family of examples showing that no Krasnosel'skii number exists for orthogonal polygons starhaped via staircase n-paths. *J. Geometry*, 94:1–6, 2009.
7 A. M. Bruckner and J. B. Bruckner. On L_n sets, the Hausdorff metric, and connectedness. *Proc. Amer. Math. Soc.*, 13:765–767, 1962.
8 H. T. Croft, K. J. Falconer, and R. K. Guy. *Unsolved problems in geometry, Volume II.* Springer, 1990.

9 P. Erdős and G. Purdy. Extremal problems in discrete geometry. In R. L. Graham, M. Grötschel, and L. Lovász, editors, *Handbook of Combinatorics*. North Holland, 1995.

10 A. Horn and F. A. Valentine. Some properties of L-sets in the plane. *Duke Math. J.*, 16:131–140, 1949.

11 T. Jech. *Set theory: The third Milennium edition*. Springer, 2003.

12 M. A. Krasnoselskii. Sur un critère pour qu'un domain soit Étoilé. *Math. Sb.*, 19:309–310., 1946.

13 A. Levy. *Basic set theory*. Springer-Verlag, Berlin, 1979.

14 E. Magazanik and M. A. Perles. Generalized convex kernels of simply connected L_n sets in the plane. *Israel J. Math.*, 160:157–171, 2007.

15 B. Peterson. Is there a Krasnonsel'skii theorem for finitely starlike sets? In M. Breen and David C. Kay, editors, *Convexity and related combinatorial geometry*. Marcel Dekker, New York, 1982.

16 Y. Shlezinger. Krasnoselskii numbers for finitely starlike sets, m.sc. thesis, hebrew university, 2011.

17 F. A. Valentine. Local convexity and L_n sets. *Proc. Amer. Math. Soc.*, 16:1305–1310, 1965.

On Guillotine Separable Packings for the Two-Dimensional Geometric Knapsack Problem

Arindam Khan
Indian Institute of Science, Bangalore, India

Arnab Maiti
Indian Institute of Technology, Kharagpur, India

Amatya Sharma
Indian Institute of Technology, Kharagpur, India

Andreas Wiese
Universidad de Chile, Santiago, Chile

Abstract

In two-dimensional geometric knapsack problem, we are given a set of n axis-aligned rectangular items and an axis-aligned square-shaped knapsack. Each item has integral width, integral height and an associated integral profit. The goal is to find a (non-overlapping axis-aligned) packing of a maximum profit subset of rectangles into the knapsack. A well-studied and frequently used constraint in practice is to allow only packings that are guillotine separable, i.e., every rectangle in the packing can be obtained by recursively applying a sequence of edge-to-edge axis-parallel cuts that do not intersect any item of the solution. In this paper we study approximation algorithms for the geometric knapsack problem under guillotine cut constraints. We present polynomial time $(1+\varepsilon)$-approximation algorithms for the cases with and without allowing rotations by 90 degrees, assuming that all input numeric data are polynomially bounded in n. In comparison, the best-known approximation factor for this setting is $3+\varepsilon$ [Jansen-Zhang, SODA 2004], even in the cardinality case where all items have the same profit.

Our main technical contribution is a structural lemma which shows that any guillotine packing can be converted into another *structured* guillotine packing with almost the same profit. In this packing, each item is completely contained in one of a constant number of boxes and **L**-shaped regions, inside which the items are placed by a simple greedy routine. In particular, we provide a clean sufficient condition when such a packing obeys the guillotine cut constraints which might be useful for other settings where these constraints are imposed.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis

Keywords and phrases Approximation Algorithms, Multidimensional Knapsack, Guillotine Cuts, Geometric Packing, Rectangle Packing

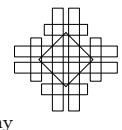
Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.48

Related Version *Full Version*: https://arxiv.org/abs/2103.09735

Acknowledgements The authors would like to thank Madhusudhan Reddy for helpful discussions. A part of this work was done when Arnab Maiti and Amatya Sharma were Narendra undergraduate summer interns at Indian Institute of Science.

1 Introduction

Geometric packing problems have many important applications in cutting stock [27], VLSI design [32], logistics [13], smart-grids [25], etc. Two-dimensional geometric knapsack (2GK), a multidimensional generalization of the classical knapsack problem, is one of the central problems in this area. We are given a set of n axis-aligned (open) rectangles (also called *items*) $I := \{1, 2, \ldots, n\}$, where rectangle i has integral width $w(i)$, integral height $h(i)$




and an associated integral profit $p(i)$. We are also given an axis-aligned square knapsack $K := [0, N] \times [0, N]$, where $N \in \mathbb{N}$. The goal is to select a subset of items $I' \subseteq I$ of maximum total profit $p(I') := \sum_{i \in I'} p(i)$ so that they can be packed in the knapsack. The packing needs to be *axis-parallel* and *non-overlapping*, i.e., such packing maps each rectangle $i \in I'$ to a new translated open rectangle $R(i) := (left(i), right(i)) \times (bottom(i), top(i))$ where $right(i) = left(i) + w(i)$, $top(i) = bottom(i) + h(i)$, $left(i) \geq 0, bottom(i) \geq 0$, $right(i) \leq N, top(i) \leq N$ and for any $i, j \in I'$, we must have $R(i) \cap R(j) = \emptyset$. In 2GK, items are not allowed to be rotated. There is another variant with rotations that we denote by 2GK(R), where items are allowed to be rotated by 90 degrees.

2GK has rich connections with many important problems, such as maximum independent set of rectangles (MISR) [2], 2-D bin packing [7], strip packing [23, 30], mixed packing [39], fair allocation [45], storage allocation [42], unsplittable flow [28], etc. Leung et al. [40] showed that the problem is strongly NP-hard. Jansen and Zhang [35] gave $(2 + \varepsilon)$-approximation algorithms for both 2GK and 2GK(R), where $\varepsilon > 0$ is an arbitrarily small constant. Finally, Gálvez et al. [24] broke the barrier of 2 by giving a 1.89-approximation algorithm for 2GK and $(3/2 + \varepsilon)$-approximation algorithm for 2GK(R). Furthermore, if the input data is quasi-polynomially bounded (i.e., $N \leq n^{(\log n)^c}$ for some $c > 0$) then there exists a quasi-polynomial time approximation scheme (QPTAS) for both problems [3]. Polynomial time approximation schemes (PTASs) are known for many special cases: if all items are small [20], if all items are squares [31, 34], if the profit of each item equals its area [5], and if we allow resource augmentation (i.e., the size of the knapsack can be slightly increased) [21, 33]. However, it is an open problem to construct a PTAS, even with pseudo-polynomial running time.

One can view geometric packing as a cutting problem where we are given a large sheet or stock unit (maybe metal, glass, wood, rubber, or cloth), which should be cut into pieces out of the given input set. Cutting technology often only allows axis-parallel end-to-end cuts called *guillotine cuts*. See [8, 49] for practical applications and software related to guillotine packing. In this setting, we seek for solutions in which we can cut out the individual objects by a recursive sequence of guillotine cuts that do not intersect any item of the solution. The related notion of *k-stage packing* was originally introduced by Gilmore and Gomory [27]. Here each stage consists of either vertical or horizontal guillotine cuts (but not both). On each stage, each of the sub-regions obtained on the previous stage is considered separately and can be cut again by using horizontal or vertical guillotine cuts. In k-stage packing, the number of cuts to obtain each rectangle from the initial packing is at most k, plus an additional cut to trim (i.e., separate the rectangles itself from a waste area). Intuitively, this means that in the cutting process we change the orientation of the cuts $k - 1$ times. The case where $k = 2$, usually referred to as *shelf packing*, has been studied extensively.

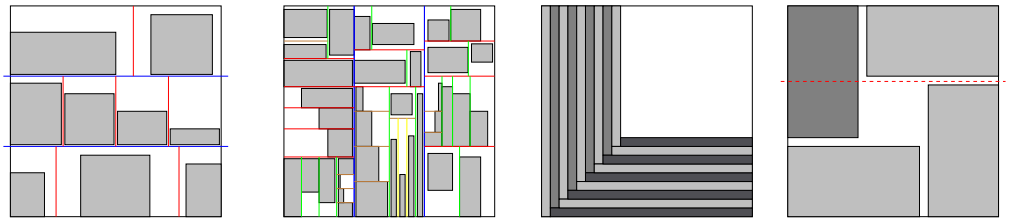


Figure 1 The first three packing are guillotine separable packings of 2-stages, 5-stages, and many stages, respectively. The last packing is not a guillotine packing as any end-to-end cut in the knapsack intersects at least one of the packed rectangles.

In this paper, we study the two-dimensional knapsack problem under guillotine cuts (2GGK). The input is the same as for 2GK, but we require additionally that the items in the solution can be separated by a sequence of guillotine cuts, and we say that then they are *guillotine separable*. NP-hardness of 2GGK follows from a reduction from the (one-dimensional) knapsack problem. Christofides et al. [14] studied the problem in 1970s. Since then many heuristics have been developed to efficiently solve benchmark instances, based on tree-search [50], branch-and-bound [29], dynamic optimization [9], tabu search [4], genetic algorithms [44], etc. Despite a staggering number of recent experimental papers [10, 15, 16, 18, 19, 22, 41, 51], there was little theoretical progress for 2GGK, due to limitations of past techniques. Since 2004, the $(3+\varepsilon)$-approximation for 2GK by Jansen and Zhang [35] has been the best-known approximation algorithm for 2GGK. Recently, Abed et al. [1] have studied approximation algorithms for the cardinality cases of 2GGK and 2GGK(R) and have given a QPTASs, assuming the input data to be quasi-polynomially bounded.

Most algorithms for 2GK utilize a *container packing* (see Section 2) which arranges the items in the knapsack such that they are packed inside a constant number of axis-aligned boxes (containers). The best sizes and locations of these containers can be guessed efficiently since there are only a constant number of them. Then inside each container the items are packed either in one-stage packings or in two-stage packings (if items are small). However, Gálvez et al. [24] show that one cannot obtain a better approximation ratio than 2 with container-based packings with only $O(1)$ many containers, due to interaction between *horizontal* (wide and thin) and *vertical* (tall and narrow) items. To break this barrier, they use a *corridor-decomposition* where the knapsack is divided into axis-parallel polygonal regions called *corridors* with constant number of regions called *subcorridors*. Vertical (resp. horizontal) items are packed in only vertical (resp. horizontal) subcorridors. After simplifying the interaction between vertical and horizontal items, they define two types of packings. In one packing, they *process* the subcorridors to obtain a container-based packing. In the other, a profitable subset of long horizontal and long vertical items are packed in an **L**-shaped region. They prove that the best of these two packings achieves a better approximation ratio than 2. However, it is not clear how to use this approach for 2GGK: even if we start with an optimal guillotine packing, the rearrangements of items may not preserve guillotine separability, and hence they might not lead to a feasible solution to 2GGK.

1.1 Our contribution

In this paper, we obtain $(1+\varepsilon)$-approximation algorithms with pseudo-polynomial running time for both 2GGK and 2GGK(R), i.e., the running time is a polynomial if the (integral) input numbers are all polynomially bounded in n. The key idea is to show that there are $(1+\varepsilon)$-approximate solutions in which the knapsack is divided into simple *compartments* that each have the shape of a rectangular box or an **L**, see Figure 2. Inside each compartment, the items are placed in a very simple way, e.g., all horizontal items are simply stacked on top of each other, all vertical items are placed side by side, and all small items are packed greedily with the Next-Fit-Decreasing-Height algorithm [17], see Figure 2. To establish this structure, we crucially exploit that the optimal solution is guillotine separable; in particular, in 2GK (where the optimal solution might not be guillotine separable) more complicated compartments may be necessary for near-optimal solutions, e.g., with the form of a ring.

While the items in our structured solution are guillotine separable, we cannot separate the compartments by guillotine cuts since we cannot cut out an **L**-shaped compartment with such cuts. This makes it difficult to compute a solution of this type since it is not sufficient to ensure that (locally) within each compartment the items are guillotine separable (which is

immediately guaranteed by our simple packings inside them). Therefore, our compartments have an important additional property: they can be separated by a *pseudo guillotine cutting sequence*. This is a cutting sequence in which each step is either a guillotine cut, or a cut along two line segments that separates a rectangular area into an **L**-shaped compartment and a smaller rectangular area, see Figure 5. We prove a strong property for compartments that admit such a pseudo guillotine cutting sequence: we show that if we pack items into such compartments in the simple way mentioned above, this will *always* yield a solution that is *globally* guillotine separable. This property and our structural result might have applications also in other settings where we are interested in solutions that are guillotine separable.

Our strong structural result allows us to construct algorithms (for the cases with and without rotations) that are relatively simple: we first guess the constantly many compartments in the structured solution mentioned above. Then we compute up to a factor $1+\varepsilon$, the most profitable set of items that can be placed nicely into them, using a simplified version of a recent algorithm in [26]. The resulting solutions use up to $\Theta(\log(nN))$ stages (unlike e.g., solutions of the Next-Fit-Decreasing-Height algorithm [17] that need only two stages). We prove a lower bound, showing that there is a family of instances of 2GGK that does not admit $(2-\varepsilon)$-approximate solutions with only $o(\log N)$-stages.

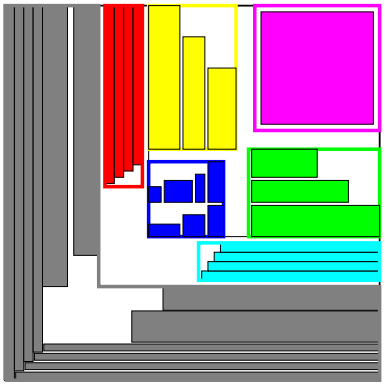


Figure 2 A structured packing of items into compartments that each have the shape of an **L**- or a rectangular box.

1.2 Other related work

There are many well-studied geometric packing problems. In the 2D bin packing problem (2BP), we are given a set of rectangular items and unit square bins, and the goal is to pack all the items into a minimum number of bins. The problem is APX-hard [6] and the currently best known approximation ratio is 1.405 [7]. In the 2D strip packing problem (2SP), we are given a set of rectangular items and a fixed-width unbounded-height strip, and the goal is to pack all the items into the strip such that the height of the strip is minimized. Kenyon and Rémila gave an APTAS for the problem [36] using a 3-stage packing.

Both 2BP and 2SP are well-studied in the guillotine setting [46]. Caprara [11] gave a 2-stage $T_\infty(\approx 1.691)$-approximation for 2BP. Afterwards, Caprara et al. [12] gave an APTAS for 2-stage 2BP and 2-stage 2SP. Later, Bansal et al. [8] showed an APTAS for guillotine 2BP. Bansal et al. [7] conjectured that the worst-case ratio between the best guillotine 2BP and the best general 2BP is 4/3. If true, this would imply a $(\frac{4}{3}+\varepsilon)$-approximation algorithm for 2BP. Seiden et al. [47] gave an APTAS for guillotine 2SP. Both the APTAS for guillotine

2BP and guillotine 2SP are based on the fact that general guillotine 2BP or guillotine 2SP can be approximated arbitrarily well by $O(1)$-stage packings, and such $O(1)$-stage packings can be found efficiently. Interestingly, we showed that this property is not true for 2GGK.

Pach and Tardos [43] conjectured that, for any set of n non-overlapping axis-parallel rectangles, there is a guillotine cutting sequence separating $\Omega(n)$ of them. Recently, the problem has received attention in [1,38] since, if true, this would imply a $O(1)$-approximation for the Maximum Independent Set of Rectangles problem, a long-standing open problem.

2 Methodology

For simplicity of presentation, we primarily focus on the cardinality case, i.e., assume that $p(i) = 1$ for each item $i \in I$. For a detailed description of the generalization to arbitrary item profits, see [37]. For each $n \in \mathbb{N}$ we define $[n] := \{1, 2, \ldots, n\}$.

We classify the input items according to their heights and widths. For two constants $1 \geq \varepsilon_{large} > \varepsilon_{small} > 0$ to be defined later, we classify each item $i \in I$ as:

- *Large*: $w_i > \varepsilon_{large} N$ and $h_i > \varepsilon_{large} N$;
- *Small*: $w_i \leq \varepsilon_{small} N$ and $h_i \leq \varepsilon_{small} N$;
- *Horizontal*: $w_i > \varepsilon_{large} N$ and $h_i \leq \varepsilon_{small} N$;
- *Vertical*: $h_i > \varepsilon_{large} N$ and $w_i \leq \varepsilon_{small} N$;
- *Intermediate:* Either $\varepsilon_{large} N \geq h_i > \varepsilon_{small} N$ or $\varepsilon_{large} N \geq w_i > \varepsilon_{small} N$.

Using standard shifting arguments, one can show that we can ignore intermediate items.

▶ **Lemma 1** ([24]). *Let $\varepsilon > 0$ and $f(.)$ be any positive increasing function such that $f(x) < x$ $\forall x \in (0, 1]$. Then we can efficiently find $\varepsilon_{large}, \varepsilon_{small} \in \Omega_\varepsilon(1)$, with $\varepsilon \geq f(\varepsilon) \geq \varepsilon_{large} \geq f(\varepsilon_{large}) \geq \varepsilon_{small}$ so that the total profit of intermediate rectangles is at most $\varepsilon p(OPT)$.*

We define *skewed* items to be items that are horizontal or vertical. Let I_{large}, I_{small}, $I_{hor}, I_{ver}, I_{skew}$ be the set of large, small, horizontal, vertical, and skewed rectangles, respectively. The corresponding intersections with OPT (the optimal guillotine packing) defines the sets OPT_{large}, OPT_{small}, OPT_{hor}, OPT_{ver}, OPT_{skew}, respectively.

2.1 Compartments

Our goal is to partition the knapsack into compartments, such that there is an $(1 + \varepsilon)$-approximate solution whose items are placed in a structured way inside these compartments. We will use two types of compartments: *box-compartments* and **L**-*compartments*.

▶ **Definition 1** (Box-compartment). *A* box-compartment B *is an axis-aligned rectangle that satisfies $B \subseteq K := [0, N] \times [0, N]$.*

▶ **Definition 2** (**L**-compartment). *An* **L**-*compartment L is a subregion of K bounded by a simple rectilinear polygon with six edges $e_0, e_1, \ldots, e_5$ such that for each pair of horizontal (resp. vertical) edges e_i, e_{6-i} with $i \in \{1, 2\}$ there exists a vertical (resp. horizontal) line segment ℓ_i of length less than $\varepsilon_{large} N/2$ such that both e_i and e_{6-i} intersect ℓ_i but no other edges intersect ℓ_i.*

Since the length of the line segments ℓ_i is less than $\varepsilon_{large} N/2$, this implies that inside an **L**-compartment L we cannot place large items, inside the horizontal part of L we cannot place vertical items, and inside the vertical part of L we cannot place horizontal items.

We seek for a structured packing inside of these compartments according to the following definitions. Inside box-compartments, we want only one type of items and we want that the skewed items are placed in a very simple way, see Figure 2.

▶ **Definition 3.** *Let B be a box-compartment and let $I_B \subseteq I$ be a set of items that are placed non-overlappingly inside B. We say that the placement of I_B is* nice *if the items in I_B are guillotine separable and additionally*

- *I_B contains only one item, or*
- *$I_B \subseteq I_{hor}$ and the items in I_B are stacked on top of each other inside B, or*
- *$I_B \subseteq I_{ver}$ and the items in I_B are placed side by side inside B, or*
- *$I_B \subseteq I_{small}$ and for each item $i \in I_B$ it holds that $w_i \leq \varepsilon \cdot w(B)$ and $h_i \leq \varepsilon \cdot h(B)$*

Inside **L**-compartments we allow only skewed items and we want them to be placed in a similar way as in the boxes, see Figure 2 and 3.

▶ **Definition 4.** *Let L be an* **L***-compartment and let $I_L \subseteq I$ be a set of items that are placed non-overlappingly inside L. We say that the placement of I_L is* nice *if*

- *$I_L \subseteq I_{skew}$, and*
- *the items in $I_L \cap I_{hor}$ are stacked on top of each other inside L, and*
- *the items in $I_L \cap I_{ver}$ are stacked side by side inside L.*

A nice placement inside an **L**-compartment yields a guillotine separable packing.

▶ **Lemma 2.** *Consider a set of items $I_L \subseteq I$ that is placed nicely inside an* **L***-compartment L. Then I_L is guillotine separable.*

Proof sketch. One can show that there always exists a guillotine cut that separates one or more horizontal or vertical items in I_L from the other items in I_L, see Figure 3. Then this argument is applied recursively. ◀

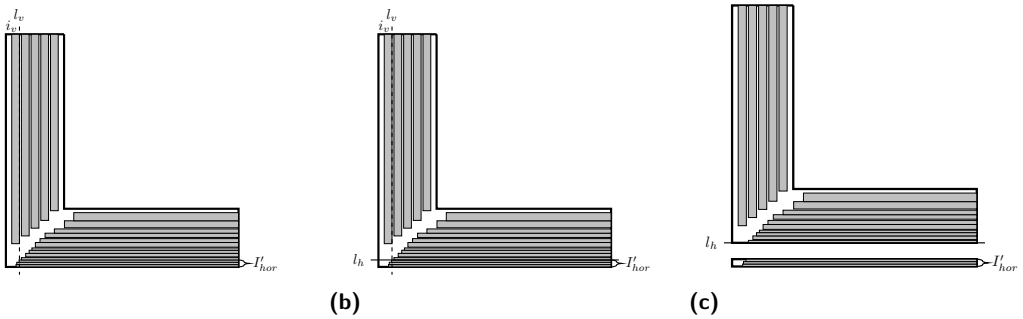


Figure 3 (a) A nicely packed set of skewed items inside an **L**-compartment. The vertical cut l_v separates the leftmost vertical item i_v from the other vertical items but it intersects the horizontal items in I'_{hor}. (b) However, then the horizontal cut l_h separates the items in I'_{hor} from the other horizontal items without intersecting any vertical item. (c) The corresponding guillotine cut that partitions the **L**-compartment into a box-compartment and a smaller **L**-compartment.

2.2 Pseudo-guillotine separable compartments

We seek to partition the knapsack into box- and **L**-compartments and then place items into these compartments. We also want to ensure that the resulting solution is guillotine separable. We could guarantee this if there was a guillotine cutting sequence that separates all compartments and require that the items inside the compartments are placed nicely. Then, we could first separate all compartments by the mentioned cutting sequence and then separate the items inside of each compartment by guillotine cuts (as they are packed nicely).

However, there is no guillotine cutting sequence that cuts out an **L**-compartment from the knapsack since no guillotine cut can separate the **L**-compartment from the area at the "inner" part of the **L**-compartment. Therefore, we require for the compartments in our knapsack only that there is a *pseudo-guillotine cutting sequence*. A pseudo-guillotine cutting sequence has the following two operations (see Figure 5): given a rectangle $R \subseteq K$ it

- applies a horizontal or vertical guillotine cut that separates R into two disjoint rectangles R_1, R_2 and then continues recursively with R_1 and R_2, or
- for an **L**-compartment $L \subseteq R$ such that $R \setminus L$ is a rectangle, it partitions R into L and $R \setminus L$ and then continues recursively with $R \setminus L$ (but not with L). Note that we cannot do this operation with every **L**-compartment $L' \subseteq R$ since possibly $R \setminus L'$ is not a rectangle.

We formalize this in the following definition.

▶ **Definition 5.** *A* pseudo-guillotine cutting sequence *(for compartments) for a set of compartments $\mathcal{C}$ is a binary tree $T = (V, E)$ where for each vertex $v \in V$ there is an associated shape $S_v \subseteq K$ such that*

- *for the root $r \in V$ of T it holds that $S_r = K$,*
- *for each internal vertex v with children u, w it holds that*
  - *S_v is a rectangle with $S_v = S_u \dot{\cup} S_w$ (so in particular S_u and S_w are disjoint),*
  - *either S_u and S_w are both rectangles or one of them is an **L**-compartment and the other is a rectangle,*
- *for each compartment $C \in \mathcal{C}$ there is a leaf $v \in V$ such that $S_v = C$.*

Observe that each **L**-compartment corresponds to a leaf node in T.

Now the important insight is that if a set of compartments $\mathcal{C}$ admits a pseudo-guillotine cutting sequence, then *any* nice placement of items inside these compartments is guillotine separable (globally). In particular, given such compartments $\mathcal{C}$, we can place items inside the compartments in $\mathcal{C}$ without needing to worry whether the resulting packing will be guillotine separable globally, as long as we place these items nicely. Intuitively, this is true since we can use the cuts of the pseudo-guillotine cutting sequence as a template for a global cutting sequence for the items: whenever the former sequence

- makes a guillotine cut, we simply do the same cut,
- when it separates an **L**-compartment L from a rectangular region R, we separate the items inside L by a sequence of guillotine cuts; it turns out that we can do this since all items inside L are placed nicely and skewed.

Finally, we separate the items inside each box-compartment B by guillotine cuts, using the fact that the items inside B are placed nicely.

▶ **Lemma 3.** *Let $\mathcal{C}$ be a set of compartments inside K that admit a pseudo-guillotine cutting sequence. Let $I' \subseteq I$ be a set of items that are placed nicely inside the compartments in $\mathcal{C}$. Then there is a guillotine cutting sequence for I'.*

Proof sketch. Let P denote the pseudo-guillotine cutting sequence. We construct a guillotine cutting sequence for I' based on P. We follow the cuts of P. Whenever P makes a guillotine cut, then we also do this guillotine cut. When P separates an **L**-compartment L from a rectangular region R, then we apply a sequence of guillotine cuts that step by step separates all items in L from $R \setminus L$. Since inside L the items are placed nicely, one can show that there exist such cuts that don't intersect any item in $R \setminus L$ (see Figure 4). ◀

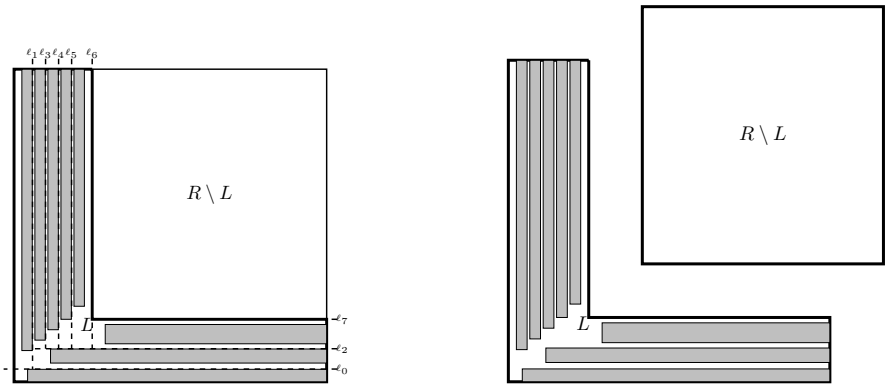


Figure 4 Partition of rectangle R into L and $R \setminus L$ when items inside L are packed nicely. $\ell_0, \ldots, \ell_7$ (dashed lines) are a sequence of guillotine cuts that ultimately separate out the items in L from R.

2.3 Near-optimal structured solutions

Our main technical contribution is to show that there exists a $(1+\varepsilon)$-approximate solution whose items can be placed nicely inside a set of compartments $\mathcal{C}$ that admit a pseudo-guillotine cutting sequence. By Lemma 3 there is a guillotine cutting sequence for them.

▶ **Lemma 4.** *There exists a set $OPT' \subseteq I$ and a partition of K into a set of $O_\varepsilon(1)$* [1] *compartments $\mathcal{C}$ such that*

- $|OPT'| \geq (1-\varepsilon)|OPT|$,
- *the compartments $\mathcal{C}$ admit a pseudo-guillotine cutting sequence,*
- *the items in OPT' can be placed nicely inside the compartments $\mathcal{C}$.*

We will prove Lemma 4 in Section 3. Our main algorithm works as follows. First, we guess the $O_\varepsilon(1)$ compartments $\mathcal{C}$ due to Lemma 4 in time $(nN)^{O_\varepsilon(1)}$ (note that we can assume w.l.o.g. that they have integral coordinates). Then we place items nicely inside $\mathcal{C}$ while maximizing the cardinality of the placed items. For this we use a $(1+\varepsilon)$-approximation algorithm which is a slight adaptation of a recent algorithm in [26] for the 2GK problem (i.e., without requiring that the computed solution is guillotine separable). In fact, we simplify some steps of that algorithm since our compartments are very simple.

▶ **Lemma 5.** *Given a set of compartments $\mathcal{C}$. In time $(nN)^{O_\varepsilon(1)}$ we can compute a set of items $ALG \subseteq I$ that are placed nicely inside $\mathcal{C}$ such that $|ALG| \geq (1-\varepsilon)|OPT'|$ for any set of items OPT' that can be placed nicely inside the compartments $\mathcal{C}$. Inside each compartment $C \in \mathcal{C}$ the set ALG admits an $O_\varepsilon(\log(nN))$-stage packing.*

We will prove Lemma 5 in Section 4. Then, Lemmas 3, 4, and 5 imply our main theorem for the cardinality case. Due to Lemma 4, our pseudo-guillotine cutting sequence has $O_\varepsilon(1)$ leaf nodes and each of them is either a box- or an **L**-compartment. The packing algorithm due to Lemma 5 gives a $O_\varepsilon(\log(nN))$-stage packing inside each compartment. This yields globally a $O_\varepsilon(\log(nN))$-stage packing. For the analysis of our $(1+\varepsilon)$-approximation algorithm for the weighted case, see [37].

[1] The notation $O_\varepsilon(f(n))$ means that the implicit constant hidden by the big O notation can depend on ε.

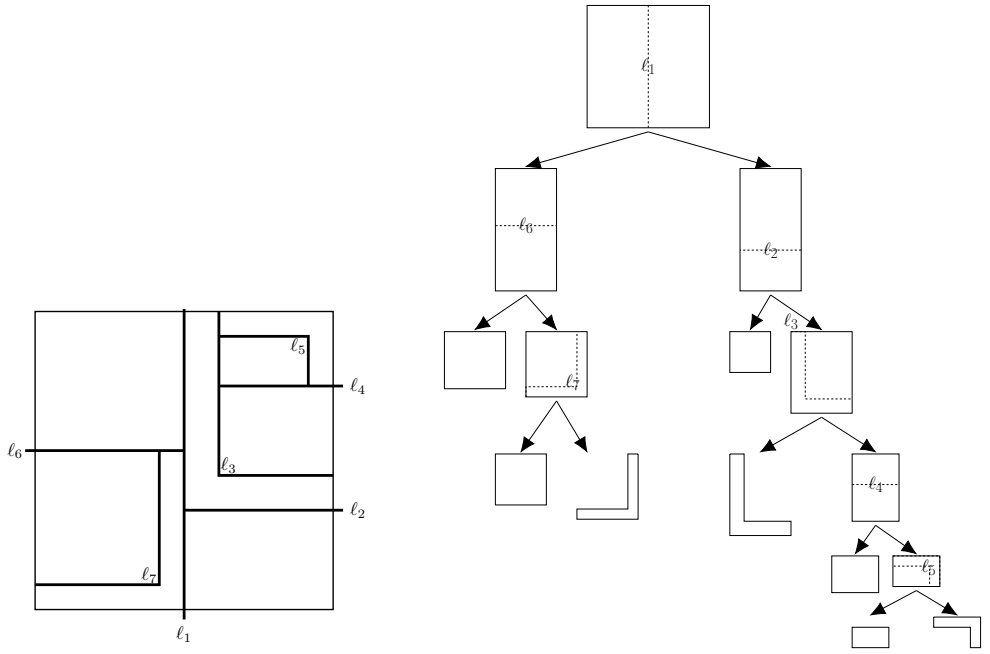


Figure 5 (a) A pseudo-guillotine cutting sequence. The first cut is l_1, and then the resulting right piece is further subdivided by ℓ_2, ℓ_3, ℓ_4 and ℓ_5. Similarly, ℓ_6, ℓ_7 subdivide the left piece. Note that ℓ_3, ℓ_5 and ℓ_7 are not guillotine cuts, but they cut out the corresponding **L**-compartments. (b) step by step pseudo-guillotine cutting sequence corresponding to Figure (a). Dashed line at each level indicates a partition of a rectangle into two regions (two boxes, or one box and one **L**-shaped).

▶ **Theorem 6.** *There is a $(1+\varepsilon)$-approximation algorithm for 2GGK with a running time of $(nN)^{O_\varepsilon(1)}$ that computes an $O_\varepsilon(\log(nN))$-stage packing.*

We obtain a similar result also for the rotational case: our structural result from Lemma 4 still holds and the algorithm due to Lemma 5 needs only some minor modifications.

▶ **Theorem 7.** *There is a $(1+\varepsilon)$-approximation algorithm for 2GGK(R) with a running time of $(nN)^{O_\varepsilon(1)}$ that computes an $O_\varepsilon(\log(nN))$-stage packing.*

3 Existence of near-optimal structured solutions

In this section, we prove Lemma 4 in the cardinality case, i.e., there exists a $(1+\varepsilon)$-approximate solution whose items can be placed nicely inside a set of compartments $\mathcal{C}$ that admit a pseudo-guillotine cutting sequence. Note that Lemma 4 trivially holds if $|OPT| \leq O_\varepsilon(1)$ and hence we assume that $|OPT|$ is larger than any given constant (thus we can drop any set of $O_\varepsilon(1)$ items from OPT while losing only a factor of $1+\varepsilon$).

Consider an optimal solution OPT and a corresponding guillotine cutting sequence S. Temporarily, we remove from the packing the items in OPT_{small}; we will put back most of them later. We identify a set of cuts of S as follows. Let ℓ_0 denote the first cut of S. Assume w.l.o.g. that ℓ_0 is vertical. If the distance of ℓ_0 to the left *and* to the right edge of K is at least $\varepsilon_{large}N/4$ then we stop. Otherwise ℓ_0 cuts K into two rectangles R_1, R_2 and assume w.l.o.g. that the width of R_1 is at most $\varepsilon_{large}N/4$. Now we consider how S continues within R_2. We continue recursively. Assume inductively that we identified a set of cuts $\ell_0, ..., \ell_{k-1}$ of S and suppose that ℓ_{k-1} is vertical cut with distance less than $\varepsilon_{large}N/4$ to the left or the

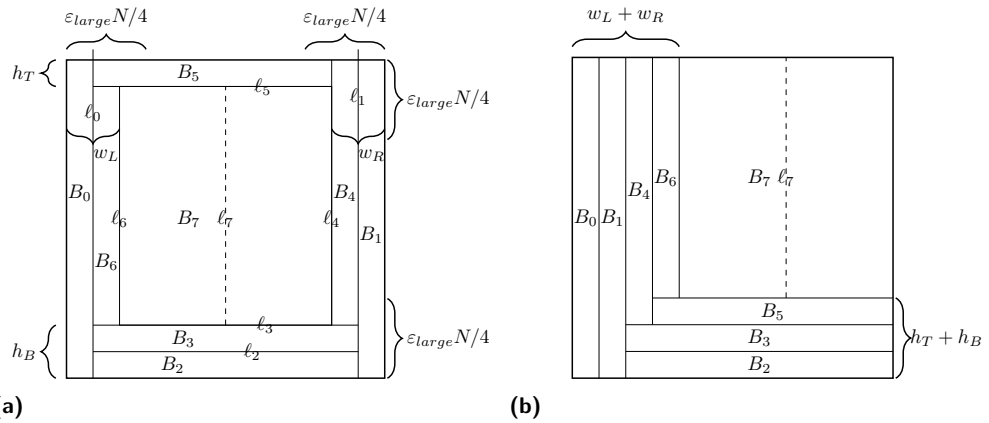


Figure 6 Transformation to obtain an **L**-compartment.

right edge of K, or that ℓ_{k-1} is horizontal cut with distance less than $\varepsilon_{large}N/4$ to the top or the bottom edge of K. Assume w.l.o.g. that ℓ_{k-1} is vertical with distance less than $\varepsilon_{large}N/4$ to the left edge of K. Then the cut ℓ_{k-1} yields two rectangles R_1, R_2, and assume that R_1 lies on the left of R_2. Then we define ℓ_k to be the next cut of S within R_2. If the distance of ℓ_k to the top *and* the bottom edge of K is at least $\varepsilon_{large}N/4$ then we stop. Otherwise we continue iteratively. Eventually, this procedure must stop, let $\ell_0, ..., \ell_k$ denote the resulting sequence. Let $B_0, ..., B_{k-1}$ denote the rectangles that are cut off by $\ell_0, ..., \ell_{k-1}$ and into which we did *not* recurse when we defined $\ell_1, ..., \ell_k$. Let B_k denote the rectangle that is cut by ℓ_k. Then each rectangle B_i with $i \in \{1, ..., k-1\}$ satisfies that $w(B_i) \leq \varepsilon_{large}N/4$ or $h(B_i) \leq \varepsilon_{large}N/4$ and in particular cannot contain both horizontal and vertical items. Also, the items of OPT inside B_i are guillotine separable. The important insight is that we can rearrange the rectangles $B_0, ..., B_k$ (while moving their items accordingly) such that $B_0, ..., B_{k-1}$ lies in an **L**-compartment $L \subseteq K$ such that $K \setminus L$ is a rectangle, i.e., L lies at the boundary of K as shown in the Figure 6.

▶ **Lemma 8.** *There exists an* **L***-compartment $L \subseteq K$ such that $K \setminus L$ is a rectangle and we can rearrange the rectangles $B_0, ..., B_k$ such that*

- *$B_0, ..., B_{k-1}$ fit non-overlappingly into L,*
- *there is a guillotine cutting sequence for $B_0, ..., B_{k-1}$,*
- *B_k fits into $K \setminus L$.*

Proof. Following the cutting sequence S as described, let us assume that $B_k := [w_L, N - w_R] \times [h_B, N - h_T]$, where $0 \leq w_R, w_L, h_T, h_B \leq \varepsilon_{large}N/4$. Therefore, the cuts $\ell_1, ..., \ell_{k-1}$ separate of a ring-like region $Q := ([0, w_L] \times [0, N]) \cup ([N - w_R, N] \times [0, N]) \cup ([0, N] \times [0, h_B]) \cup ([0, N] \times [N - h_T, h_T])$ (see Figure 6). Note that some of the values w_R, w_L, h_T, h_B might be 0. The rectangles $B_0, ..., B_{k-1}$ fit in Q and we want to show that we can rearrange the rectangles in $B_0, ..., B_{k-1}$ into an **L**-compartment $L \subseteq K$ such that $L := ([0, w_L + w_R] \times [0, N]) \cup ([0, N] \times [0, h_B + h_T])$ and there is a guillotine cutting sequence for $B_0, ..., B_{k-1}$. Clearly, B_k fits into $K \setminus L$. We prove the claim by induction on k. The base case is trivial. W.l.o.g. assume the vertical cut ℓ_0 that divides K into B_0, R', where B_0 lies on the left of R'. Hence, $B_0 := [0, b_0] \times [0, N]$ and $R' := K \setminus B_0$. We use induction on R' to find a packing of $B_1, ..., B_{k-1}$ in $L' := [b_0, w_L + w_R] \times [0, N] \cup [0, N] \times [0, h_B + h_T]$. Therefore, adding B_0 to L' yields the desired **L**-compartment L. For the guillotine cutting sequence

for $B_0, ..., B_{k-1}$, we follow ℓ_0 and afterwards the guillotine cutting sequence for $B_1, ..., B_{k-1}$ obtained by induction from R'. The other cases, i.e., when B_0 lies right or top or bottom of R', follow analogously. ◀

We adjust the packing of OPT according to Lemma 8, i.e., for each rectangle B_i with $i \in \{0, ..., k\}$ we move its items according to where B_i was moved due to the lemma. The resulting packing inside L might not be nice. However, we can fix this by dropping at most $O_\varepsilon(1)$ items and subdividing L into $O_\varepsilon(1)$ box-compartments and a smaller **L**-compartment $L' \subseteq L$ that *lies at the outer boundary of* L, i.e., such that $L \setminus L'$ is again an **L**-compartment and $h(L') = h(L)$ and $w(L') = w(L)$.

▶ **Lemma 9.** *Given an* **L***-compartment* L *containing a set of items* $I(L)$*. There exists a partition of* L *into one* **L***-compartment* $L' \subseteq L$ *and* $O_\varepsilon(1)$ *box-compartments* $\mathcal{B}(L)$ *such that*

- L' *lies at the outer boundary of* L*,*
- *the box-compartments in* $\mathcal{B}(L)$ *are guillotine separable, and*
- *there is a nice placement of a set of items* $I'(L) \subseteq I(L)$ *with* $|I'(L)| \geq (1-\varepsilon)|I'(L)| - O_\varepsilon(1)$ *inside* $\mathcal{B}(L)$ *and* L'*.*

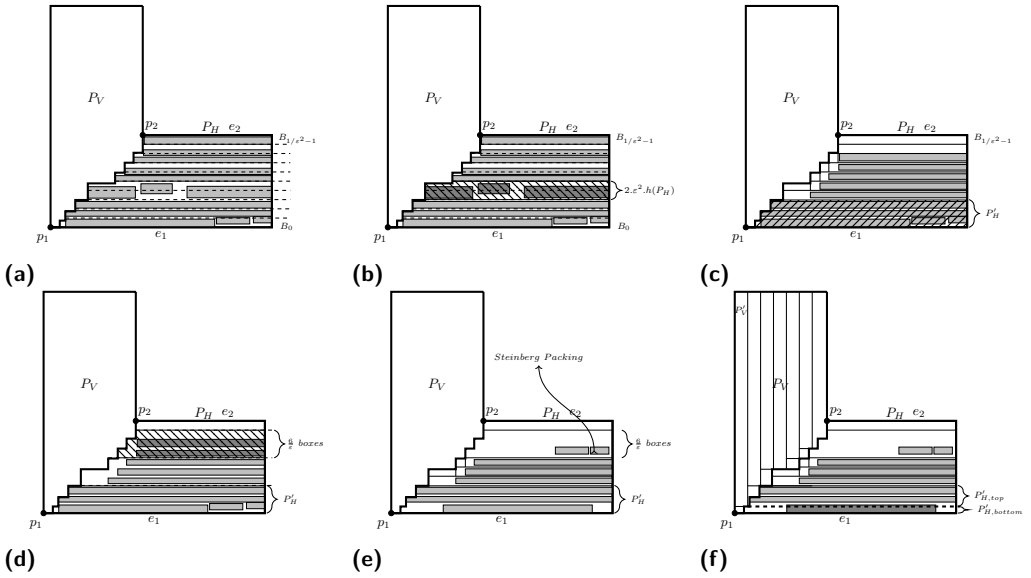


Figure 7 Processing done in Lemma 9 to obtain a nice packing in L-compartment.

Proof sketch. Since it is sufficient to place $(1-\varepsilon)|I'(L)| - O_\varepsilon(1)$ items, we can drop $O_\varepsilon(1)$ items. So w.l.o.g. assume that $I(L)$ contains only skewed items (i.e., we remove all large items). Intuitively, we partition L into two polygons P_H and P_V that are separated via a monotone axis-parallel curve connecting the two vertices of L at the bend of L, such that P_H contains all horizontal items placed inside L and P_V contains all vertical items inside L, see Figure 7a. We rearrange the items in P_H and P_V separately, starting with P_H. Denote by $I(P_H) \subseteq I(L)$ the items of $I(L)$ placed inside P_H.

We place $1/\varepsilon^2$ boxes inside P_H of height $\varepsilon^2 h(P_H)$ each, stacked one on top of the other. We define their width maximally large such that they are still contained inside P_H (note that some area of P_H is then not covered by these boxes), see Figure 7b. Denote by $\{B_0, ..., B_{1/\varepsilon^2-1}\}$ these boxes in this order, such that B_0 touches the longer horizontal edge of P_H. With a shifting argument, we can show that there are two consecutive boxes B_{j^*}, B_{j^*+1} with

$j^* \leq 1/\varepsilon$ that intersect with at most an $O_\varepsilon(1) + O(\varepsilon|I(P_H)|)$ items in $I(P_H)$. We remove these items. Let $P'_H \subseteq P_H$ denote the part of P_H underneath B_{j^*} (see Figure 7c). We move down by $\varepsilon^2 h(P)$ units each item in $I(P_H)$ that intersect one of the boxes $B_{j^*+2}, ..., B_{1/\varepsilon^2-1}$ and we remove all $O_\varepsilon(1)$ items from $I(L)$ that intersect more than one box. Note that then the moved items fit into the boxes $\mathcal{B}' := \{B_{j^*+1}, ..., B_{1/\varepsilon^2-2}\}$.

Using another shifting step, we delete all items in $6/\varepsilon$ consecutive boxes of $\mathcal{B}'$; since there are $\Omega(1/\varepsilon^2)$ boxes in $\mathcal{B}'$ this costs only a factor $1 + O(\varepsilon)$ in the profit. We use the empty space to place in it all items in P'_H that are shorter than the shorter horizontal edge of P_H, see Figure 7e. One can show that they can be placed into this empty space using Steinberg's algorithm [48] (maintaining guillotine separability) since the available space is much larger than the area of the items to be placed. For the remaining items in P'_H one can show that the width of each of them is more than half of the width of L. Hence, we can assume w.l.o.g. that they are placed nicely within P'_H. Again, we remove all items that intersect more than one box after this movement, which are at most $O_\varepsilon(1)$ items. Denote by $\mathcal{B}_{hor}$ the resulting set of boxes.

We do a symmetric procedure for P_V, yielding a set of boxes $\mathcal{B}_{ver}$ and a nicely packed region P'_V. Intuitively, we want to define L' as $P'_H \cup P'_V$. However, $P'_H \cup P'_V$ might not have exactly the shape of an **L**-compartment. Nevertheless, one can show that we can subdivide one of these polygons, say P'_H, along a horizontal line into two subpolygons $P'_{H,\text{top}}, P'_{H,\text{bottom}}$ (with $P'_{H,\text{top}}$ lying on the top of $P'_{H,\text{bottom}}$) such that

- we can place the items in $P'_{H,\text{top}}$ into another set of $O_\varepsilon(1)$ boxes $\mathcal{B}'_{hor}$ that are non-overlapping with $\mathcal{B}_{hor} \cup \mathcal{B}_{ver}$, and
- $L' := P'_{H,\text{bottom}} \cup P'_V$ forms an **L**-compartment, see Figure 7f.

Then the items are nicely placed inside L'. To each of the $O_\varepsilon(1)$ boxes $B \in \mathcal{B}_{hor} \cup \mathcal{B}'_{hor} \cup \mathcal{B}_{ver}$ we apply a standard routine that removes some of the items inside B and partitions B into smaller boxes, such that the remaining items inside these smaller boxes are nicely placed. ◀

Therefore, we define that the first cuts of our pseudo-guillotine cutting sequence S' looks as follow: we first separate K into L' and $K \setminus L'$ and then separate the boxes in $\mathcal{B}(L)$. Then we apply a guillotine cut to the rectangular area $K \setminus L$ that corresponds to ℓ_k (since we moved the items in B_k we need to adjust ℓ_k accordingly), which yields two rectangular areas R_1, R_2. With each of them we continue recursively, i.e., we apply the same routine that we had applied to K above.

We do not recurse further if for a considered rectangular area R it holds that $h(R) < \varepsilon_{large} N$ or $w(R) < \varepsilon_{large} N$. In this case R contains only horizontal or only vertical items, respectively. However, these items might not be packed nicely. Thus, we apply to R a similar routine as in Lemma 9. In a sense, R behaves like a degenerate **L**-compartment with only four edges. Also note that R is a box-compartment.

▶ **Lemma 10** ([37]). *Given a box-compartment B containing a set of items $I(B)$ with $h(B) < \varepsilon_{large} N$ or $w(B) < \varepsilon_{large} N$, there exists a partition of B into $O_\varepsilon(1)$ box-compartments $\mathcal{B}(B)$ such that*

- *the box-compartments in $\mathcal{B}(B)$ are guillotine separable, and*
- *there is a nice placement of a set of items $I'(B) \subseteq I(B)$ with $|I'(B)| \geq (1-\varepsilon)|I'(B)| - O_\varepsilon(1)$ inside $\mathcal{B}(B)$.*

It remains to put back the (small) items in OPT_{small}. Intuitively, we assign them to the empty space in our $O_\varepsilon(1)$ constructed compartments. More formally, we subdivide our compartments further into smaller compartments by guillotine cuts, some of the resulting

compartments are empty, and into those we assign the small items with the Next-Fit-Decreasing-Height algorithm [17]. For each of these compartments we ensure that their height and width are $\varepsilon_{small}N/\varepsilon$. There might be empty space that is not used in this way, however, we can ensure that its total area is very small, e.g., at most $O(\varepsilon^2 N^2)$. This allows us to pack essentially all items in OPT_{small} (handling a few special cases differently, e.g., if the total area of the items in OPT_{small} is very small).

Let S' denote the resulting pseudo-guillotine cutting sequence. We need to argue that this yields in total $O_\varepsilon(1)$ compartments. This follows easily since every time we identify a sequence of cuts $\ell_0, ..., \ell_k$ of S, we construct exactly one **L**-compartment and $O_\varepsilon(1)$ box-compartments. Also, after each such operation, we recurse on rectangular areas R_1, R_2 that are at least by $\varepsilon_{large}N/4$ units thinner or shorter (i.e., by at least $\varepsilon_{large}N/4$ units smaller in one of the two dimensions) than the rectangular area that we had started with when we constructed $\ell_0, ..., \ell_k$ (which is the whole knapsack K in the first iteration). Also, when we do not recurse further we subdivide the remaining region into $O_\varepsilon(1)$ box-compartments. Each resulting compartment is subdivided into $O_\varepsilon(1)$ smaller compartments when we place the small items. Hence, the depth of the binary tree T defining the pseudo-guillotine cutting sequence S' is $O_\varepsilon(1)$ and thus we define at most $O_\varepsilon(1)$ compartments in total. In particular, we applied Lemmas 9 and 10 at most $O_\varepsilon(1)$ times and, therefore, the constructed solution contains at least $(1-\varepsilon)|OPT| - O_\varepsilon(1)$ items. A refined argument extends this to the weighted case as well, we refer the readers to the full version [37] for a detailed description.

4 Assigning items into compartments

For proving Lemma 5, we need to provide an algorithm that, given a set of compartments $\mathcal{C}$, computes a solution $ALG \subseteq I$ with $p(ALG) \geq (1-\varepsilon)p(OPT')$ that can also be placed nicely in $\mathcal{C}$ (where $OPT' \subseteq I$ is the subset of I of maximum profit that can be placed nicely in the compartments in $\mathcal{C}$).

First, we guess for each box-compartment $B \in \mathcal{C}$ which case of Definition 3 applies, i.e., whether B contains only a single large item, or only horizontal items, or only vertical items, or only small items. For each box-compartment $B \in \mathcal{C}$ for which we guessed that it contains only one large item, we simply guess this item. We can do this deterministically in time $O(n^{|\mathcal{C}|}) = n^{O_\varepsilon(1)}$ for all such box-compartments $B \in \mathcal{C}$.

Then, for assigning the small items, we use a standard reduction to the Generalized Assignment Problem (GAP) [24] for selecting a near-optimal set of small items and an assignment of these items into the corresponding box-compartments. Inside of each box-compartment B we place the items with the Next-Fit-Decreasing-Height algorithm [17] which results in a 2-stage guillotine separable packing for the items inside B.

▶ **Lemma 11** ([37]). *Given a set of box compartments $\mathcal{B}$ such that a set of items $I^*_{small} \subseteq I_{small}$ can be placed non-overlappingly inside $\mathcal{B}$, in $n^{O_\varepsilon(1)}$ time we can we can compute a set of items $I'_{small} \subseteq I_{small}$ with $p(I'_{small}) \geq (1-\varepsilon)p(I^*_{small})$ and a nice placement of the items in I'_{small} inside $\mathcal{B}$ which is guillotine separable with $O_\varepsilon(1)$ stages.*

Let $\mathcal{C}_{skew} \subseteq \mathcal{C}$ denote the compartments in $\mathcal{C}$ into which skewed items are placed in OPT' (which in particular contains all **L**-compartments in $\mathcal{C}$). It remains to select a profitable set of items from I_{skew} that can be placed nicely in the compartments in $\mathcal{C}_{skew}$. For this task, we use a recent algorithm in [26] which is a routine for 2GK which takes as input (in our terminology) a set of box- and **L**-compartments, and also compartments of more general shapes (e.g., with the shapes of a U or a Z). In time $(nN)^{O_\varepsilon(1)}$, it computes a

subset of the input items of maximum total profit, up to a factor of $1+\varepsilon$, that can be placed non-overlappingly inside the given compartments. In fact, it first partitions the given compartments such that there exists a profitable solution for the smaller compartments inside of which the items are placed nicely (according to our definition). Then it computes a $(1+\varepsilon)$-approximation of the most profitable subset of items that can be placed nicely.

In our setting, we can skip the first step since in OPT' the items are already placed nicely inside the compartments $\mathcal{C}_{skew}$. Hence, we execute directly the second part the algorithm in [26]. In fact, a simpler version of that routine is sufficient since we have only box- and **L**-compartments. The algorithm in [26] can handle also the case where rotations by 90 degree are allowed, and the same holds for the routine in Lemma 11. Thus our result works for the case with rotations as well. We refer to the full version [37] for a complete and self-contained description of this routine, adapted to the guillotine setting. In particular, inside each compartment its solution is guillotine separable with $O_\varepsilon(\log nN)$ stages.

5 Power of stages in guillotine packing

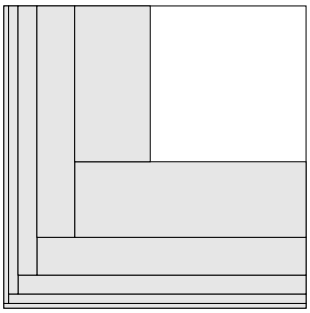


Figure 8 Hard example for Theorem 12.

Our two algorithms compute packings with $O_\varepsilon(\log(nN))$-stages. This raises the question whether one can obtain $(1+\varepsilon)$-approximate solutions with fewer stages. In particular, for the related guillotine 2BP and guillotine 2SP problems there are APTASs whose solutions use $O(1)$-stage packings [8, 47]. However, we show that in contrast for 2GGK sometimes $\Omega(\log N)$ stages are necessary already for a better approximation ratio than 2, even if there are only skewed items. For a detailed proof, we refer to the full version [37].

▶ **Theorem 12.** *For any constant $0 < \varepsilon < \frac{1}{2}$, there is a family of instances of 2GGK with only skewed items for which any $(2-\varepsilon)$-approximate solution requires $k = \Omega(\varepsilon \log N)$ stages.*

6 Conclusion

Two main open questions are to obtain PTASes for 2GGK and 2GK. We conjecture that the worst-case ratio between the optimal profit of 2GGK and 2GK is 4/3. If this conjecture is true, then a PTAS for 2GGK will imply a $4/3+\varepsilon$-approximation for 2GK, improving the present best approximation guarantee [24].

References

1 Fidaa Abed, Parinya Chalermsook, José R. Correa, Andreas Karrenbauer, Pablo Pérez-Lantero, José A. Soto, and Andreas Wiese. On guillotine cutting sequences. In *APPROX*, pages 1–19, 2015.

2 Anna Adamaszek and Andreas Wiese. Approximation schemes for maximum weight independent set of rectangles. In *FOCS*, pages 400–409, 2013.

3 Anna Adamaszek and Andreas Wiese. A quasi-PTAS for the two-dimensional geometric knapsack problem. In *SODA*, pages 1491–1505, 2015.

4 Ramón Alvarez-Valdés, Antonio Parajón, and José Manuel Tamarit. A tabu search algorithm for large-scale guillotine (un) constrained two-dimensional cutting problems. *Computers & Operations Research*, 29(7):925–947, 2002.

5 Nikhil Bansal, Alberto Caprara, Klaus Jansen, Lars Prädel, and Maxim Sviridenko. A structural lemma in 2-dimensional packing, and its implications on approximability. In *ISAAC*, pages 77–86, 2009.

6 Nikhil Bansal, Jose R Correa, Claire Kenyon, and Maxim Sviridenko. Bin packing in multiple dimensions: inapproximability results and approximation schemes. *Mathematics of Operations Research*, 31:31–49, 2006.

7 Nikhil Bansal and Arindam Khan. Improved approximation algorithm for two-dimensional bin packing. In *SODA*, pages 13–25, 2014.

8 Nikhil Bansal, Andrea Lodi, and Maxim Sviridenko. A tale of two dimensional bin packing. In *FOCS*, pages 657–666, 2005.

9 J. E. Beasley. An exact two-dimensional non-guillotine cutting tree search procedure. *Operations Research*, 33(1):49–64, 1985.

10 István Borgulya. An eda for the 2d knapsack problem with guillotine constraint. *Central European Journal of Operations Research*, 27(2):329–356, 2019.

11 Alberto Caprara. Packing 2-dimensional bins in harmony. In *FOCS*, pages 490–499, 2002.

12 Alberto Caprara, Andrea Lodi, and Michele Monaci. Fast approximation schemes for two-stage, two-dimensional bin packing. *Mathematics of Operations Research*, 30(1):150–172, 2005.

13 Henrik I. Christensen, Arindam Khan, Sebastian Pokutta, and Prasad Tetali. Approximation and online algorithms for multidimensional bin packing: A survey. *Computer Science Review*, 24:63–79, 2017.

14 Nicos Christofides and Charles Whitlock. An algorithm for two-dimensional cutting problems. *Operations Research*, 25(1):30–44, 1977.

15 François Clautiaux, Ruslan Sadykov, François Vanderbeck, and Quentin Viaud. Combining dynamic programming with filtering to solve a four-stage two-dimensional guillotine-cut bounded knapsack problem. *Discrete Optimization*, 29:18–44, 2018.

16 François Clautiaux, Ruslan Sadykov, François Vanderbeck, and Quentin Viaud. Pattern-based diving heuristics for a two-dimensional guillotine cutting-stock problem with leftovers. *EURO Journal on Computational Optimization*, 7(3):265–297, 2019.

17 Edward G. Coffman, Jr, Michael R. Garey, David S. Johnson, and Robert E. Tarjan. Performance bounds for level-oriented two-dimensional packing algorithms. *SIAM Journal on Computing*, 9:808–826, 1980.

18 Alessandro Di Pieri. Algorithms for two-dimensional guillotine packing problems. Master's thesis, University of Padova, Italy, 2013.

19 Mohammad Dolatabadi, Andrea Lodi, and Michele Monaci. Exact algorithms for the two-dimensional guillotine knapsack. *Computers & Operations Research*, 39(1):48–53, 2012.

20 Aleksei V. Fishkin, Olga Gerber, and Klaus Jansen. On efficient weighted rectangle packing with large resources. In *ISAAC*, pages 1039–1050, 2005.

21 Aleksei V. Fishkin, Olga Gerber, Klaus Jansen, and Roberto Solis-Oba. Packing weighted rectangles into a square. In *MFCS*, pages 352–363, 2005.

22 Fabio Furini, Enrico Malaguti, and Dimitri Thomopulos. Modeling two-dimensional guillotine cutting problems via integer programming. *INFORMS Journal on Computing*, 28(4):736–751, 2016.

23 Waldo Gálvez, Fabrizio Grandoni, Afrouz Jabal Ameli, Klaus Jansen, Arindam Khan, and Malin Rau. A tight (3/2+ϵ) approximation for skewed strip packing. In *APPROX/RANDOM*, pages 44:1–44:18, 2020.

24 Waldo Gálvez, Fabrizio Grandoni, Sandy Heydrich, Salvatore Ingala, Arindam Khan, and Andreas Wiese. Approximating geometric knapsack via l-packings. In *FOCS*, pages 260–271, 2017.

25 Waldo Gálvez, Fabrizio Grandoni, Salvatore Ingala, and Arindam Khan. Improved pseudo-polynomial-time approximation for strip packing. In *FSTTCS*, pages 9:1–9:14, 2016.

26 Waldo Gálvez, Fabrizio Grandoni, Arindam Khan, Diego Ramirez-Romero, and Andreas Wiese. Improved approximation algorithms for 2-dimensional knapsack: Packing into multiple l-shapes, spirals and more. In *To appear in SoCG*, 2021.

27 P. C. Gilmore and Ralph E. Gomory. Multistage cutting stock problems of two and more dimensions. *Operations research*, 13(1):94–120, 1965.

28 Fabrizio Grandoni, Tobias Mömke, Andreas Wiese, and Hang Zhou. A $(5/3 + \varepsilon)$-approximation for unsplittable flow on a path: placing small tasks into boxes. In *STOC*, pages 607–619, 2018.

29 Eleni Hadjiconstantinou and Nicos Christofides. An exact algorithm for general, orthogonal, two-dimensional knapsack problems. *European Journal of OR*, 83(1):39–56, 1995.

30 Rolf Harren, Klaus Jansen, Lars Prädel, and Rob van Stee. A $(5/3 + \epsilon)$-approximation for strip packing. *Computational Geometry*, 47(2):248–267, 2014.

31 Sandy Heydrich and Andreas Wiese. Faster approximation schemes for the two-dimensional knapsack problem. In *SODA*, pages 79–98, 2017.

32 Dorit S. Hochbaum and Wolfgang Maass. Approximation schemes for covering and packing problems in image processing and VLSI. *Journal of the ACM*, 32(1):130–136, 1985.

33 Klaus Jansen and Roberto Solis-Oba. New approximability results for 2-dimensional packing problems. In *MFCS*, pages 103–114, 2007.

34 Klaus Jansen and Roberto Solis-Oba. A polynomial time approximation scheme for the square packing problem. In *IPCO*, pages 184–198, 2008.

35 Klaus Jansen and Guochuan Zhang. On rectangle packing: maximizing benefits. In *SODA*, pages 204–213, 2004.

36 Claire Kenyon and Eric Rémila. A near-optimal solution to a two-dimensional cutting stock problem. *Mathematics of Operations Research*, 25(4):645–656, 2000.

37 Arindam Khan, Arnab Maiti, Amatya Sharma, and Andreas Wiese. On guillotine separable packings for the two-dimensional geometric knapsack problem, 2021. `arXiv:2103.09735`.

38 Arindam Khan and Madhusudhan Reddy Pittu. On guillotine separability of squares and rectangles. In *APPROX/RANDOM*, pages 47:1–47:22, 2020.

39 Arindam Khan, Eklavya Sharma, and K. V. N. Sreenivas. Approximation algorithms for generalized multidimensional knapsack. *CoRR*, abs/2102.05854, 2021.

40 Joseph Y. T. Leung, Tommy W. Tam, C. S. Wong, Gilbert H. Young, and Francis Y. L. Chin. Packing squares into a square. *Journal of Parallel and Distributed Computing*, 10(3):271–275, 1990.

41 Andrea Lodi, Michele Monaci, and Enrico Pietrobuoni. Partial enumeration algorithms for two-dimensional bin packing problem with guillotine constraints. *Discrete Applied Mathematics*, 217:40–47, 2017.

42 Tobias Mömke and Andreas Wiese. Breaking the barrier of 2 for the storage allocation problem. In *ICALP*, pages 86:1–86:19, 2020.

43 János Pach and Gábor Tardos. Cutting glass. In *SoCG*, pages 360–369, 2000.

44 V. Parada, R. Munoz, and A. Gomes. A hybrid genetic algorithm for the two-dimensional cutting problem. *Evolutionary algorithms in management applications. Springer, Berlin*, pages 183–196, 1995.

45 Deval Patel, Arindam Khan, and Anand Louis. Group fairness for knapsack problems. In *To appear in AAMAS*, 2021.

46 Enrico Pietrobuoni. *Two-dimensional bin packing problem with guillotine restrictions*. PhD thesis, University of Bologna, Italy, 2015.

47 Steven S. Seiden and Gerhard J. Woeginger. The two-dimensional cutting stock problem revisited. *Mathematical Programming*, 102(3):519–530, 2005.

48 A. Steinberg. A strip-packing algorithm with absolute performance bound 2. *SIAM Journal on Computing*, 26(2):401–409, 1997.

49 Paul E. Sweeney and Elizabeth Ridenour Paternoster. Cutting and packing problems: a categorized, application-orientated research bibliography. *Journal of the Operational Research Society*, 43(7):691–706, 1992.

50 K. V. Viswanathan and A. Bagchi. An exact best-first search procedure for the constrained rectangular guillotine knapsack problem. In *AAAI*, pages 145–149, 1988.

51 Lijun Wei and Andrew Lim. A bidirectional building approach for the 2d constrained guillotine knapsack packing problem. *European Journal of Operational Research*, 242(1):63–71, 2015.

Restricted Constrained Delaunay Triangulations

Marc Khoury
University of California at Berkeley, CA, USA

Jonathan Richard Shewchuk
University of California at Berkeley, CA, USA

Abstract

We introduce the restricted constrained Delaunay triangulation (restricted CDT), a generalization of both the restricted Delaunay triangulation and the constrained Delaunay triangulation. The restricted CDT is a triangulation of a surface whose edges include a set of user-specified constraining segments. We define the restricted CDT to be the dual of a restricted Voronoi diagram defined on a surface that we have extended by topological surgery. We prove several properties of restricted CDTs, including sampling conditions under which the restricted CDT contains every constraining segment and is homeomorphic to the underlying surface.

2012 ACM Subject Classification Theory of computation → Randomness, geometry and discrete structures

Keywords and phrases restricted Delaunay triangulation, constrained Delaunay triangulation, surface meshing, surface reconstruction, topological surgery, portals

Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.49

Related Version *Complete version*: `https://people.eecs.berkeley.edu/~jrs/papers/rcdt.pdf`

Funding Supported in part by the National Science Foundation under Awards CCF-1423560 and CCF-1909204 and in part by a National Science Foundation Graduate Research Fellowship.

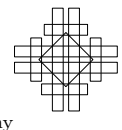
Acknowledgements This work was initiated at the Workshop on Geometric Algorithms in the Field hosted by the Lorentz Center in Leiden, the Netherlands during June 2014. We thank the organizers – Sándor Fekete, Maarten Löffler, Bettina Speckmann, and Jo Wood – and the Lorentz Center for providing accommodations. We especially thank Bruno Lévy for posing the problem this paper answers, Marc van Kreveld for helpful discussions, and the referees for improving the paper.

1 Introduction

The constrained Delaunay triangulation (CDT) in the plane [19, 25, 13] is a popular geometric construction that shares some of the advantages and mathematical properties of the Delaunay triangulation, but also permits users to constrain specified edges to be part of the triangulation. CDTs are used in applications such as computer graphics, geographical information systems, and guaranteed-quality mesh generation algorithms [12]. Our goal here is to offer a mathematically rigorous way to define a Delaunay-like triangulation on a curved surface embedded in three-dimensional space, with the same ability to constrain edges.

Another variant of the Delaunay triangulation, called the *restricted Delaunay triangulation* (RDT), has become a well-established way of generating triangulations on curved surfaces [16]. RDTs have equipped theorists to rigorously prove the correctness of algorithms for surface reconstruction [14] and surface mesh generation [12]. In this paper we introduce *restricted constrained Delaunay triangulations* (restricted CDTs), which combine ideas from CDTs and RDTs to enable the enforcement of constraining edges in RDTs.

Think of the restricted CDT as a function that takes in three inputs: a compact, smooth surface $\Sigma \subset \mathbb{R}^3$ without boundary; a finite set $V \subset \Sigma$ of points (called *sites* or *vertices*); and a finite set S of line segments whose endpoints are in V. If certain conditions on the density



of V and the lengths of the segments are met then, as illustrated in Figure 1, the output is a simplicial complex $\mathcal{T}$ such that the set of vertices of $\mathcal{T}$ is V, the set of edges of $\mathcal{T}$ is a superset of S, and $\mathcal{T}$ is a triangulation of Σ. The last phrase means that the *underlying space* of $\mathcal{T}$, written $|\mathcal{T}| = \bigcup_{\tau \in \mathcal{T}} \tau$, is homeomorphic to Σ.

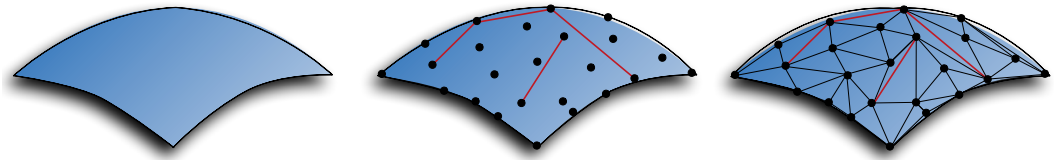


Figure 1 Given a set of points sampled from a surface Σ and a set of segments, red, we wish to compute a triangulation of Σ that contains all of the red segments.

Although Delaunay triangulations in the plane can be constrained to include arbitrary edges, the same is not true of three-dimensional Delaunay triangulations; consider the fact that not all nonconvex polyhedra can be tetrahedralized [24]. Nor is it always possible to constrain arbitrary edges to be part of a surface triangulation. Our challenge is to establish conditions on the input that guarantee that a suitable triangulation exists.

We follow the example of the RDT, which is defined by dualizing a *restricted Voronoi diagram*. Given inputs Σ and V (but no segments), the *restricted Voronoi cell* of a site $v \in V$, denoted $\mathrm{Vor}|_\Sigma v$, is the set of all points on Σ for which v is the closest site in V (possibly tied for closest), as measured by the Euclidean distance in $\mathbb{R}^3$. Equivalently, $\mathrm{Vor}|_\Sigma v = \mathrm{Vor}\, v \cap \Sigma$, where $\mathrm{Vor}\, v$ is v's standard Voronoi cell in $\mathbb{R}^3$. The name "restricted Voronoi cell" arises because $\mathrm{Vor}|_\Sigma v$ is the restriction of $\mathrm{Vor}\, v$ to the surface Σ.

A *restricted Voronoi face* is any nonempty set of points found by taking the intersection of one or more restricted Voronoi cells. The *restricted Voronoi diagram* $\mathrm{Vor}|_\Sigma V$ is the cell complex containing all the restricted Voronoi cells and faces.

The *restricted Delaunay triangulation* $\mathrm{Del}|_\Sigma V$ is the simplicial complex dual to $\mathrm{Vor}|_\Sigma V$. If the restricted Voronoi cells of two sites $v, w \in V$ have a nonempty intersection (typically a path on Σ), then vw is a *restricted Delaunay edge* in $\mathrm{Del}|_\Sigma V$. If the restricted Voronoi cells of three sites $u, v, w \in V$ have a nonempty intersection (typically a single point on Σ, called a *restricted Voronoi vertex*), then $\triangle uvw$ is a *restricted Delaunay triangle* in $\mathrm{Del}|_\Sigma V$. Every site in V is a vertex in $\mathrm{Del}|_\Sigma V$. Note that $\mathrm{Del}|_\Sigma V$ may not be a valid simplicial complex unless V is a sufficiently dense sample of Σ, perhaps with suitable perturbations of Σ and V. See Section 3 for a more nuanced discussion.

To modify RDTs so that we can constrain edges, we borrow from Seidel [25] the idea of an *extended Voronoi diagram*, which is the natural dual of the CDT in the plane. Seidel performs a topological surgery on the plane in which each segment in S becomes a slit cut in the plane; upon these slits he glues topological extensions called "secondary sheets" on which additional portions of the extended Voronoi diagram are drawn. Likewise, we perform surgery by cutting slits in the surface Σ and grafting independent new surfaces called *extrusions* onto Σ at these slits. We think of these slits as *portals*: an ant crawling on the surface across a constraining edge finds itself transported by the portal to a secondary space where the extended surface continues along an infinite extrusion.

A key contribution of this paper is our definition of the restricted constrained Delaunay triangulation, as the dual of the Voronoi diagram restricted to this surgically extended surface. Another contribution is to prove several properties of restricted CDTs, including conditions under which the restricted CDT contains every constraining edge, conditions under which the restricted CDT is homeomorphic to the underlying surface Σ, and a characterization of which vertices must be considered to compute the triangles near a segment.

Shewchuk [26] demonstrates that for Delaunay mesh generators that create high-quality meshes of domains in the plane with constraining segments, the use of a CDT (rather than a pure Delaunay triangulation) reduces the number of triangles and vertices – on some domains, by as much as 25%. He also proves that there is a theoretical advantage: Delaunay meshing with a CDT offers a guarantee of a "size optimal" mesh with no angle less than 26.56°, whereas an *unconstrained* Delaunay triangulation offers a weaker guarantee, a size optimal mesh with no angle less than 20.7°. It is very likely that surface meshing algorithms based on restricted CDTs can offer the same advantages, compared to what pure RDTs can achieve.

An alternative approach sometimes suggested is to define a Voronoi diagram based on an intrinsic (geodesic) distance metric, then obtain a triangulation by duality. This idea is mathematically elegant, but computing a geodesic Voronoi diagram entails numerical approximation algorithms [18, 20, 21], which add coding complexity and running time. RDTs are popular in surface mesh generation because they are easier to compute. We emphasize that although our construction of restricted CDTs may seem complicated, it is in the service of producing simple algorithms. (In particular, Theorems 1 and 3 simplify computing the triangles near a segment.) See Section 6 for some speculation on prospective algorithms.

2 Portals and topological surgery

Informally, a *portal* P is a doorway between two topological spaces, with P shared by both. Our main topological construction starts with disjoint topological spaces Y and Z, then glues them together into a single space by specifying an equivalence relationship between a subset of points $P \subset Y$ and a subset $P' \subset Z$. For clarity, we explain Seidel's construction of portals in the plane [25] first, then our construction of portals and an extended surface in $\mathbb{R}^3$.

2.1 Portals and extended Voronoi diagrams in the plane

Let $X = \mathbb{R}^2$ and let S be a finite set of line segments in the plane; the segments may intersect each other only at their endpoints. Consider a segment $s = pq \in S$ (meaning s has endpoints p and q). The relative interior of s, denoted $\operatorname{relint} s$, consists of all points on s except p and q. Let the *slitted plane* $X_s = X - \operatorname{relint} s$ be the plane with the relative interior of s removed. The affine hull of s has two "sides." Our goal is to augment X_s by gluing it to two additional topological spaces, one for each side of s, along the slit created by removing $\operatorname{relint} s$. The three spaces are glued together along two portals, each of which is topologically a copy of s. Thus an ant crawling on the extended space that crosses s from one side finds itself in a *secondary branch*; and an ant that crosses s from the other side finds itself in a different secondary branch. After repeating this augmentation for every segment in S, we can draw on the extended space an *extended Voronoi diagram* whose dual is the CDT.

Topologically, X_s has a hole such that X_s is *almost* an open set, except that X_s has two boundary points, p and q. We want to glue two additional spaces to X_s – one for each side of s – so we augment X_s with additional points that serve as two portals to those additional spaces. We define a closed topological space $\overline{X}_s$ by augmenting X_s with two connected curves ζ^+ and ζ^-, called *portals*, that together serve as the boundary of the hole. Each of ζ^+ and ζ^- has p and q as its endpoints, but the two curves share no other points. In essence, the portals are copies of s with shared endpoints. Formally, $\overline{X}_s$ is the completion of the incomplete metric space X_s with respect to the shortest-path metric in X_s.

The points in X_s inherit Cartesian coordinates from the plane, and the points on the portals ζ^+ and ζ^- inherit Cartesian coordinates from the segment s. Two points in $\overline{X}_s$ – one on ζ^+ and one on ζ^- – can have the same (x, y)-coordinate values yet be topologically distinct.

Let $\mathbb{R}^2_-$ and $\mathbb{R}^2_+$ be two copies of $\mathbb{R}^2$. We treat $\overline{X}_s$, $\mathbb{R}^2_-$, and $\mathbb{R}^2_+$ as three distinct topological spaces that all inherit the Cartesian coordinate system – so two points in two different spaces can have the same coordinate values yet be topologically distinct.

Informally, we glue $\mathbb{R}^2_+$ to $\overline{X}_s$ along ζ^+ and glue $\mathbb{R}^2_-$ to $\overline{X}_s$ along ζ^-. Formally, we write $x \equiv y$ if x and y have the same coordinate values, even though they may lie in different spaces. Let p and q be the endpoints of s. Define an equivalence relation $\sim$ as

$$x \sim y \iff \begin{cases} x = y & x, y \in \overline{X}_s \text{ or } x, y \in \mathbb{R}^2_+ \text{ or } x, y \in \mathbb{R}^2_-, \\ x \equiv y & x \in \mathbb{R}^2_+ \text{ and } y \in \zeta^+, \\ x \equiv y & x \in \mathbb{R}^2_- \text{ and } y \in \zeta^-, \\ x \equiv p \equiv y \text{ or } x \equiv q \equiv y & x \in \mathbb{R}^2_+ \text{ and } y \in \mathbb{R}^2_-. \end{cases} \tag{1}$$

With $\sim$ we construct the quotient space $\widetilde{X} = (\overline{X}_s \sqcup \mathbb{R}^2_+ \sqcup \mathbb{R}^2_-)/\sim$. We refer to $\overline{X}_s$ as the *principal branch* and refer to $\mathbb{R}^2_+$ and $\mathbb{R}^2_-$ as *secondary branches*. Figures 2 and 3 illustrate this construction. Note that in the quotient space, the endpoints p and q of the segment s are present in, and shared by, all three of the original spaces.

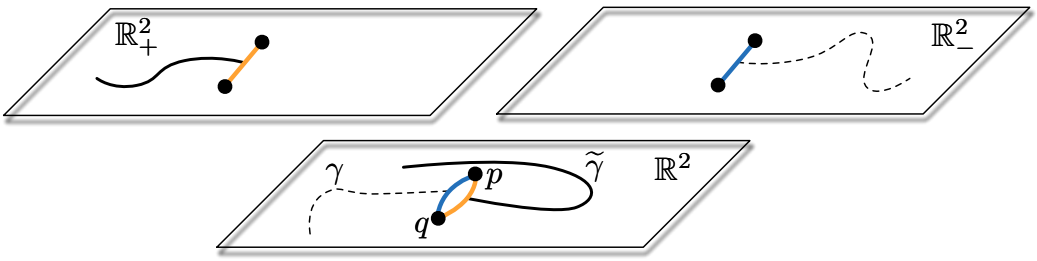


Figure 2 The completion of the slitted plane has a topological hole bounded by two portals, marked in blue and orange. (Geometrically, the two portals are straight line segments that occupy exactly the same coordinates.) The equivalence relation $\sim$ identifies the blue path in the principal branch with the blue path in $\mathbb{R}^2_-$; likewise the two orange paths become one. A path in the principal branch (bottom) that enters a portal continues in the appropriate secondary branch.

The construction works for any finite number $m = |S|$ of non-crossing segments. Let $X_S = X - \bigcup_{s \in S} \operatorname{relint} s$. Let $\overline{X}_S$ be the completion of X_S with respect to the shortest-path metric in X_S, which adds two portals for each segment. Then we construct a quotient space $\widetilde{X}$ composed of $\overline{X}_S$ and $2m$ copies of $\mathbb{R}^2$ glued along the $2m$ portals bounding the m holes in $\overline{X}_S$.

For the sake of defining the Voronoi diagram of a finite set of sites in $\widetilde{X}$, Seidel [25] defines a distance function on $\widetilde{X}$ which is essentially the Euclidean distance, except that the distance between two points is infinite if they are not visible from each other. (Note that this distance function is not a metric.) A path $\gamma \subset \widetilde{X}$ may pass through portals and visit secondary branches, but because of the slits we have cut in X_S, γ cannot cross the relative interior of a segment without being transported by a portal. We call a path *straight* if its Cartesian embedding is a straight line segment. Two points $p, q \in \widetilde{X}$ are *visible* from each other if there is a straight path $\gamma \subset \widetilde{X}$ with endpoints p and q. The distance $\widehat{d}(p, q)$ from p to q is the Euclidean distance if p and q are visible from each other; otherwise, $\widehat{d}(p, q) = \infty$.

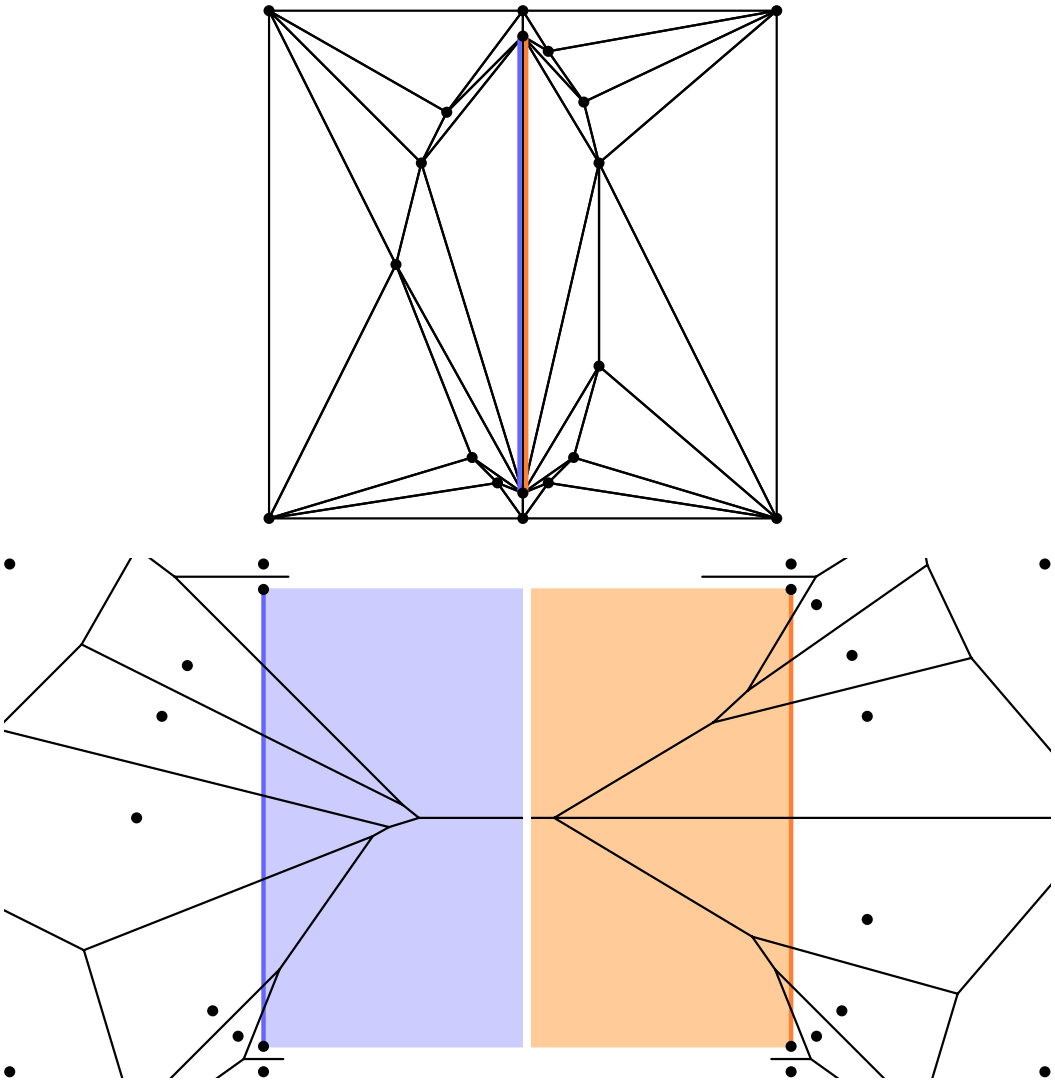


Figure 3 A one-segment CDT (top) and its dual extended Voronoi diagram (bottom). The blue and orange regions show the portions of the Voronoi diagram on the secondary branches.

The extended Voronoi diagram assigns each point in $\widetilde{X}$ to (the Voronoi cells of) one or more sites in V. Those sites must be visible from the point; no site can claim a point it cannot see. If a point on a secondary branch is claimed by a site other than the branch's portal's endpoints, the site is visible from the point through the portal, as Figure 3 illustrates. Seidel gives an algorithm for constructing the extended Voronoi diagram, and by duality the CDT.

2.2 Portals on surfaces embedded in $\mathbb{R}^3$

A similar construction works for a compact, smooth surface without boundary $\Sigma \subset \mathbb{R}^3$. However, whereas in the plane we construct one new topological space, here we will require two. We surgically augment the surface Σ by cutting slits along *portal curves*, one for each segment, and gluing two *extrusions* onto each portal curve, yielding an *extended surface* $\widetilde{\Sigma}$. The purpose of this extended surface is to serve as a canvas upon which we can draw an extended restricted Voronoi diagram, which we dualize to define a restricted CDT.

To define a Voronoi diagram we need a distance function, and $\widetilde{\Sigma}$ alone does not provide one that is easily computed. While an intrinsic (geodesic) distance might be ideal in principle, for the sake of speed and a simple implementation, we use the Euclidean distance in $\mathbb{R}^3$ as RDTs do; but we must modify the Euclidean distance so that the restricted Voronoi diagram respects the input segments. Hence most of our work will be to construct a surgically modified three-dimensional space $\widetilde{X}$ in which we embed $\widetilde{\Sigma} \subset \widetilde{X}$. Like Seidel's extended space in Section 2.1, $\widetilde{X}$ obstructs (and supports) visibility in a manner that is suitable for defining a restricted Voronoi diagram on $\widetilde{\Sigma}$ and makes it easy to compute restricted CDTs.

To define $\widetilde{X}$, we specify portals in $\mathbb{R}^3$ where points will be removed, analogous to cutting slits in the plane. Each portal is a two-dimensional ruled surface with boundary (not generally flat), approximately perpendicular to Σ. The intersection of a portal with Σ is a portal curve. Each portal has two "sides," and on each side we glue an additional copy of $\mathbb{R}^3$ to form $\widetilde{X}$. In each copy of $\mathbb{R}^3$ we embed an extrusion to form $\widetilde{\Sigma}$. The extended Voronoi diagram assigns each point x on $\widetilde{\Sigma}$ to one or more sites in V that are visible from x along straight paths in $\widetilde{X}$.

To define portal geometry, we need several definitions. The *medial axis* M of Σ is the closure of the set of all points in $\mathbb{R}^3$ for which the closest point on Σ is not unique. Intuitively, the medial axis of Σ is meant to capture the "middle" of the region bounded by Σ. A *medial ball* is a ball whose center lies on M and whose boundary intersects Σ (tangentially), but the interior of the ball does not. For any point $x \in \Sigma$, there are one or two medial balls that have x on their boundaries, called *medial balls at* x. If there are two, there is one on each side of Σ. If there is only one, it is enclosed by Σ.

For $x \in \Sigma$, the *normal line* $\mathcal{L}_x$ at x is the line orthogonal to Σ at x with $x \in \mathcal{L}_x$. The *normal segment* ℓ_x at x is a line segment or ray whose endpoints lie on M, satisfying $x \in \ell_x \subset \mathcal{L}_x$. If there are two medial balls at x, the endpoints of ℓ_x are the centers of those two medial balls. If there is only one medial ball at x, then ℓ_x is a ray originating at the medial ball's center.

The *local feature size* function is $\mathrm{lfs} : \Sigma \to \mathbb{R}$, $x \mapsto d(x, M)$ where $d(x, M)$ denotes the Euclidean distance from x to M. We require that Σ is smooth in the sense that $\inf_{x \in \Sigma} \mathrm{lfs}(x) > 0$. A finite point set $V \subset \Sigma$ is an *ϵ-sample of* Σ if for every point $x \in \Sigma$, $d(x, V) \leq \epsilon\, \mathrm{lfs}(x)$. That is, the ball with center x and radius $\epsilon\, \mathrm{lfs}(x)$ contains at least one point in V. See Figure 4.

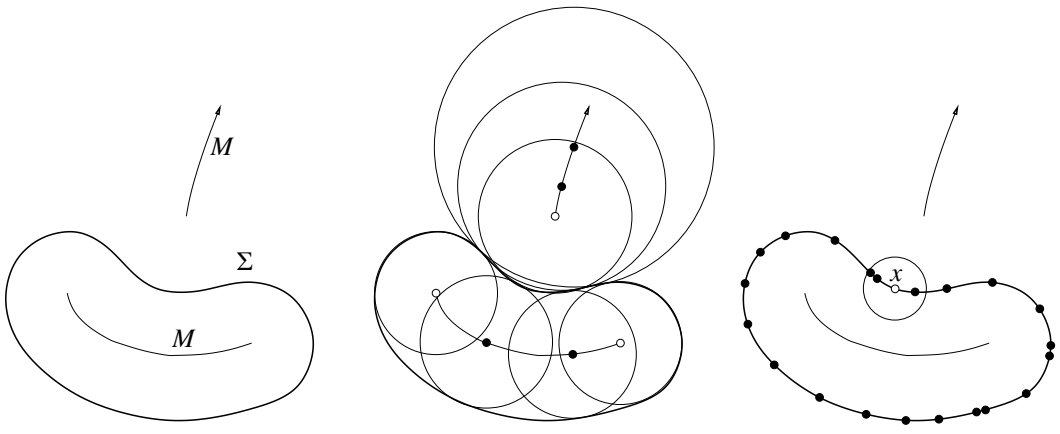


Figure 4 Left: A 1-manifold Σ and its medial axis M (as medial axes in three dimensions are hard to draw or understand). This medial axis is unbounded; one of its components extends infinitely far away. Center: Some of the medial balls that define M. Right: A 0.5-sample of Σ (filled circles). The ball with center x and radius $0.5\,\mathrm{lfs}(x)$ contains a site.

Let S be a finite set of line segments whose endpoints are in V, called the *segments*, which constrain the restricted CDT. Consider a segment $s = pq \in S$ (its endpoints are $p, q \in V$). Let B_s be the *diametric ball* of s – the smallest closed ball such that $s \subset B_s$, so that s is a diameter of B_s. Suppose that $d(p,q) \leq \rho\, \mathrm{lfs}(p)$ for some $\rho \in (0,1)$; that is, s is short relative to the local feature size. Then $B_s \cap \Sigma$ is a topological disk [12, Lemma 12.6].

Suppose that we know (or can approximate) the unit vector n_p normal to Σ at any site p. We choose a *cutting plane* $h_s \supset s$ that is locally orthogonal to the surface Σ at p or q (or perhaps somewhere between p and q). We use h_s to specify a *portal curve* $\zeta_s = h_s \cap B_s \cap \Sigma$, which is a single connected curve from p to q on Σ. There is not a canonical choice of cutting plane (and thus portal curve) for s, and the user might be presented with a range of choices, but for our presentation here, we choose $h_s = \mathrm{span}\{n_p, \vec{pq}\}$. We require that the portal curves do not cross each other. More precisely, the relative interior of a portal curve may not intersect another portal curve nor a site in V.

Our requirement that each portal curve must lie on a plane has both a theoretical motivation and a practical one. The fact that every constraining segment is an edge in the restricted CDT (Theorem 2) depends on the fact that each portal curve lies in a plane and its extrusions are orthogonal to that plane. The requirement simplifies algorithms for computing a restricted CDT, because the Voronoi cells on an extrusion are solely influenced by sites on the other side of the cutting plane – plus the segment endpoints p and q. (See Theorem 1.)

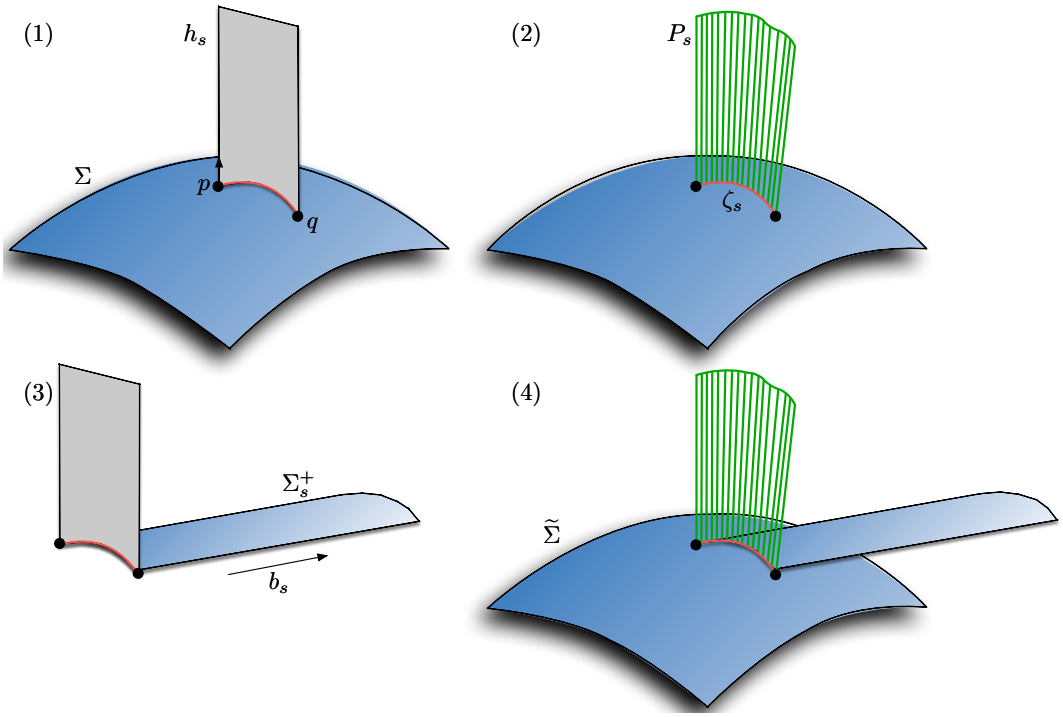


Figure 5 (1) The plane h_s intersects Σ in a curve; the portal curve ζ_s (red) is the portion of this curve in the diametric ball B_s of the segment $s = pq$. (2) Our portal P_s, shown in green, is the union of the normal segments (locally orthogonal to Σ) of the points on the portal curve ζ_s. The normal segments terminate on the medial axis M. (3) We extrude the portal curve ζ_s into $\mathbb{R}^3_+$ in a direction b_s orthogonal to h_s, thus defining Σ_s^+. (4) We glue the extrusion Σ_s^+ to $\overline{\Sigma}_S$ (the surface Σ with slits cut into it) along ζ_s^+ at the entrance to the portal P_s^+.

Figure 5 illustrates our portal construction. For each segment s, the portal $P_s = \bigcup_{x \in \zeta_s} \ell_x$ is the union of the normal segments of the points on the portal curve ζ_s. Hence a portal is a ruled surface, topologically two-dimensional but not lying in a plane. Each portal reaches to the medial axis, thereby obstructing visibility so that sites on one "side" of a segment do not influence the restricted Delaunay triangles on the other "side."

If two segments share an endpoint p, then their portals share the boundary segment ℓ_p. The other location where portals' boundaries may intersect each other is at the medial axis. However, no portal intersects the relative interior of another portal.

We construct the extended space $\widetilde{X}$ as we did in Section 2.1, with P_s replacing s and $\mathbb{R}^3$ replacing $\mathbb{R}^2$. Let $X = \mathbb{R}^3$. Let $X_S = X \setminus \bigcup_{s \in S} \operatorname{relint} P_s$, which is $\mathbb{R}^3$ with the relative interior of each portal removed. Let $\overline{X}_S$ be the completion of the incomplete metric space X_S endowed with the shortest path metric. The effect of completing X_S is to augment each "slit" P_s with two portals P_s^+ and P_s^-, one for each side of P_s. These two portals are distinct copies of P_s, but P_s^+ and P_s^- share a common boundary $\partial P_s = P_s^+ \cap P_s^- = P_s^+ \cap X_S = P_s^- \cap X_S$.

For each segment $s \in S$, let $\mathbb{R}^3_{s+}$ and $\mathbb{R}^3_{s-}$ be two topologically distinct copies of $\mathbb{R}^3$, called *secondary branches*. The points in each secondary branch and the points in the *principal branch* $\overline{X}_S$ all inherit Cartesian coordinates, but points with the same coordinates in different branches are topologically distinct. Define an equivalence relation $\sim$ analogous to (1) that identifies (glues) the points of the portal $P_s^+ \subset \overline{X}_S$ with the points in $\mathbb{R}^3_{s+}$ having the same coordinates, and identifies the points of $P_s^- \subset \overline{X}_S$ with the corresponding points in $\mathbb{R}^3_{s-}$. Thus we glue $2m$ copies of $\mathbb{R}^3$ along the $2m$ portals bounding the m holes in $\overline{X}_S$. The *extended space* is the quotient space $\widetilde{X} = (\overline{X}_S \sqcup \bigsqcup_{s \in S} \mathbb{R}^3_{s+} \sqcup \bigsqcup_{s \in S} \mathbb{R}^3_{s-})/\sim$.

Similarly, we surgically modify Σ to construct an extended surface $\widetilde{\Sigma} \subset \widetilde{X}$, as shown in the bottom two illustrations in Figure 5. Let $\Sigma_S = \Sigma \setminus \bigcup_{s \in S} \operatorname{relint} \zeta_s$ be the surface with the portal curve interiors removed, and let the *principal surface* $\overline{\Sigma}_S \subset \overline{X}_S$ be its completion. For each $s \in S$, $\overline{\Sigma}_S$ includes two portal curves $\zeta_s^+ \subset P_s^+$ and $\zeta_s^- \subset P_s^-$, one for each side of the cutting plane h_s. We extrude ζ_s^+ into $\mathbb{R}^3_{s+}$ and ζ_s^- into $\mathbb{R}^3_{s-}$, each in one of the two directions normal to the cutting plane h_s. Let b_s be a unit vector normal to h_s, directed to pass through P_s^+ from the principal branch to $\mathbb{R}^3_{s+}$. For each point $x \in \zeta_s$ we define a ray $x_s^+ = \{x + \omega b_s \in \mathbb{R}^3_{s+} : \omega \in [0, \infty)\}$ and a ray $x_s^- = \{x - \omega b_s \in \mathbb{R}^3_{s-} : \omega \in [0, \infty)\}$ (specifying points by their coordinates). We then define two *extrusions*, the ruled surfaces $\Sigma_s^+ = \{x_s^+ : x \in \zeta_s\} \subset \mathbb{R}^3_{s+}$ and $\Sigma_s^- = \{x_s^- : x \in \zeta_s\} \subset \mathbb{R}^3_{s-}$. The *extended surface* is $\widetilde{\Sigma} = (\overline{\Sigma}_S \sqcup \bigsqcup_{s \in S} \Sigma_s^+ \sqcup \bigsqcup_{s \in S} \Sigma_s^-)/\sim$.

3 Restricted constrained Delaunay triangulations

To define the restricted constrained Delaunay triangulation, we first define the *extended restricted Voronoi diagram* (or just *extended Voronoi diagram* for short) on the extended surface $\widetilde{\Sigma}$. As in Section 2.1, we define a distance function $\widehat{d}(p, q)$ that is the Euclidean distance in $\mathbb{R}^3$ if p and q are visible to each other along a straight path in $\widetilde{X}$, or ∞ if they cannot see each other. For any $v \in V$, the *extended restricted Voronoi cell* of v is

$$\operatorname{Vor}|_{\widetilde{\Sigma}} v = \{x \in \widetilde{\Sigma} : \widehat{d}(x, v) \leq \widehat{d}(x, u),\ \forall u \in V\}.$$

An *extended restricted Voronoi face* is any nonempty intersection of one or more extended restricted Voronoi cells. The *extended restricted Voronoi diagram* $\operatorname{Vor}|_{\widetilde{\Sigma}} V$ is the cell complex containing all the extended restricted Voronoi cells and faces.

We define the *restricted constrained Delaunay subdivision* $\operatorname{Del}|_{\widetilde{\Sigma}} V$ to be the polyhedral complex dual to the extended Voronoi diagram in this sense: for each extended Voronoi face $f \in \operatorname{Vor}|_{\widetilde{\Sigma}} V$, let $W \subseteq V$ be the set of sites whose restricted Voronoi cells include f and let f^* be the convex hull of W. We say that f^* is the face *dual* to f. Then $\operatorname{Del}|_{\widetilde{\Sigma}} V = \{f^* : f \in \operatorname{Vor}|_{\widetilde{\Sigma}} V\}$.

A one-point face in $\mathrm{Vor}|_{\widetilde{\Sigma}} V$ is called an *extended restricted Voronoi vertex*, and its dual is a polygonal or polyhedral face in $\mathrm{Del}|_{\widetilde{\Sigma}} V$, usually a triangle. If an intersection of two extended restricted Voronoi cells includes a path on Σ, we call it an *extended restricted Voronoi edge*, and its dual is a (straight) *restricted constrained Delaunay edge* in $\mathrm{Del}|_{\widetilde{\Sigma}} V$. Figure 6 illustrates an extended Voronoi vertex on an extrusion and its dual restricted Delaunay triangle, as well as three extended Voronoi edges and their dual restricted constrained Delaunay edges.

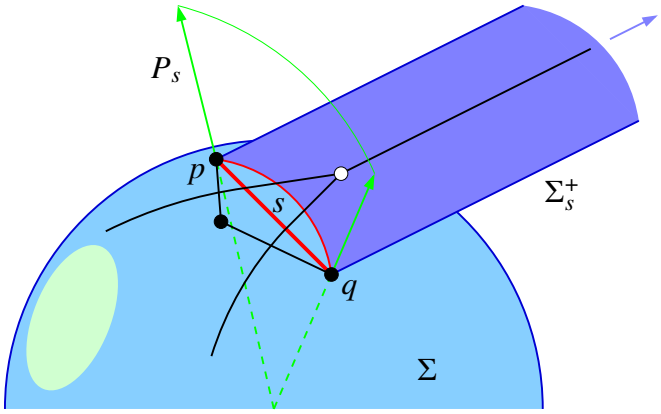


Figure 6 An extended Voronoi vertex on an extrusion and its dual restricted Delaunay triangle.

If $\mathrm{Del}|_{\widetilde{\Sigma}} V$ contains a polyhedron, we can perturb $\widetilde{\Sigma}$ infinitesimally so that $\widetilde{\Sigma}$ does not pass through the polyhedron's circumcenter; thus with suitable perturbations, $\mathrm{Del}|_{\widetilde{\Sigma}} V$ contains no polyhedra. Relatedly, just as a standard Delaunay triangulation in the plane can be ambiguous if four vertices lie on a common circle, if V has four or more cocircular vertices then $\mathrm{Del}|_{\widetilde{\Sigma}} V$ might contain polygons with four or more sides. If desired, a triangulation can be obtained by subdividing each polygonal face into triangles or by an infinitesimal perturbation of V. We recommend the former in practice, but for the sake of our proofs, we will exploit the latter. For simplicity, we will assume in this paper that no point on $\widetilde{\Sigma}$ is equidistant from four visible vertices in V; then every polygonal face is a triangle and we can call $\mathrm{Del}|_{\widetilde{\Sigma}} V$ a *restricted constrained Delaunay triangulation* (restricted CDT).

Whereas a restricted Delaunay triangulation (RDT) is a subcomplex of a three-dimensional Delaunay triangulation, we know no natural three-dimensional complex that has the restricted CDT as a subcomplex. One could define a Voronoi diagram over $\widetilde{X}$ and dualize it, but there is no reason to suppose the dual will be a valid polyhedral complex: the Voronoi cells that are supposed to be kept apart by portals are likely to meet near the medial axis. (The dual complex would also be difficult to compute.) The rest of this paper seeks sampling conditions that tame the extended Voronoi diagram (over $\widetilde{\Sigma}$, not $\widetilde{X}$) and its dual restricted CDT.

Now we present several useful properties of extended Voronoi diagrams and restricted CDTs, supposing that no segment is too long. The following theorem shows that the sites whose extended Voronoi cells lie in part on an extrusion Σ_s^+ must lie on the side of the cutting plane h_s strictly opposite Σ_s^+ (excepting the endpoints of s, which lie on h_s). Thus the restricted Voronoi vertices on Σ_s^+ dualize to restricted Delaunay triangles that are also on the side of h_s opposite Σ_s^+. This theorem simplifies computing the restricted CDT, because an algorithm only needs to look at sites in one halfspace when computing the portion of $\mathrm{Vor}|_{\widetilde{\Sigma}} V$ that lies on Σ_s^+. Unfortunately, the proof is five pages long; see the full-length article [17].

▶ **Theorem 1** (Cutting Plane Theorem). *Let $s \in S$ be a segment with endpoints $p, q \in V$ such that $d(p,q) \le \rho\,\mathrm{lfs}(p)$ for $\rho \le 0.47$. Consider a point $x \in \Sigma_s^+$ and a site $v \in V \setminus \{p,q\}$ such that $x \in \mathrm{Vor}|_{\widetilde{\Sigma}} v$. Then v is strictly on the side of h_s opposite Σ_s^+. (The symmetric claim holds for any $x \in \Sigma_s^-$.)*

The next theorem shows that the restricted CDT $\mathrm{Del}|_{\widetilde{\Sigma}} V$ contains every edge in S.

▶ **Theorem 2** (Constraint Theorem). *Let $s \in S$ be a segment with endpoints $p, q \in V$ such that $d(p,q) \le \rho\,\mathrm{lfs}(p)$ for $\rho \le 0.47$. Then $\mathrm{Vor}|_{\widetilde{\Sigma}} p \cap \mathrm{Vor}|_{\widetilde{\Sigma}} q \neq \emptyset$. Hence pq is an edge in $\mathrm{Del}|_{\widetilde{\Sigma}} V$.*

Moreover, the rays p_s^+ and p_s^- lie in the interior of $\mathrm{Vor}|_{\widetilde{\Sigma}} p$ ("interior" with respect to the space $\widetilde{\Sigma}$), and neither ray intersects any other extended restricted Voronoi cell. Likewise, q_s^+ and q_s^- lie in the interior of $\mathrm{Vor}|_{\widetilde{\Sigma}} q$, and neither ray intersects another cell.

Proof. We will show that $\mathrm{Vor}|_{\widetilde{\Sigma}} p$ meets $\mathrm{Vor}|_{\widetilde{\Sigma}} q$ on the extrusion Σ_s^+, as Figures 3 and 6 show. (The same is true on Σ_s^-.) Let Π be the plane orthogonally bisecting s. Consider the point $z = \Pi \cap \zeta_s$ on the portal curve and the ray $z_s^+ = \Pi \cap \Sigma_s^+$, whose origin is z. Let x be a point on z_s^+. Note that z is the point closest to x on the portal plane h_s, and xz is perpendicular to zp. We will show that for all $x \in z_s^+$ sufficiently far from z, $x \in \mathrm{Vor}|_{\widetilde{\Sigma}} p \cap \mathrm{Vor}|_{\widetilde{\Sigma}} q$.

Theorem 1 states that for every site $v \in V \setminus \{p, q\}$ whose extended Voronoi cell $\mathrm{Vor}|_{\widetilde{\Sigma}} v$ intersects Σ_s^+, v is strictly on the side of h_s opposite Σ_s^+. Therefore, there exists some $\delta > 0$ such that $d(x,v) \ge d(x,z) + \delta$ for *every* such site v. Consider any point $x \in z_s^+$ such that $d(x,z) \ge d(z,p)^2/(2\delta)$. By Pythagoras' Theorem, for every site v whose cell intersects Σ_s^+,

$$d(x,p)^2 = d(x,z)^2 + d(z,p)^2 \le d(x,z)^2 + 2\delta\, d(x,z) < (d(x,z) + \delta)^2 \le d(x,v)^2.$$

Hence $d(x,q) = d(x,p) < \widehat{d}(x,v)$ for every site $v \in V \setminus \{p,q\}$. As x is visible from p and q, $x \in \mathrm{Vor}|_{\widetilde{\Sigma}} p$ and $x \in \mathrm{Vor}|_{\widetilde{\Sigma}} q$. Hence $\mathrm{Vor}|_{\widetilde{\Sigma}} p \cap \mathrm{Vor}|_{\widetilde{\Sigma}} q \neq \emptyset$.

To prove the final claim, consider a point $x \in p_s^+$. Observe that p is the point nearest x on h_s and $d(x,p) < d(x,q)$. Repeating the reasoning above, there exists some $\delta > 0$ such that $d(x,v) \ge d(x,p) + \delta$ for every site $v \in V \setminus \{p\}$ such that $\mathrm{Vor}|_{\widetilde{\Sigma}} v$ intersects Σ_s^+. Therefore, there is an open neighborhood $N \subset \widetilde{\Sigma}$ of x such that $N \subset \mathrm{Vor}|_{\widetilde{\Sigma}} p$ and N intersects no other cell. The same reasoning applies to points on p_s^-, q_s^+, and q_s^-. Hence p_s^+ and p_s^- lie in the interior of $\mathrm{Vor}|_{\widetilde{\Sigma}} p$ and do not intersect another extended Voronoi cell. ◀

The shape of our extrusions Σ_s^+ and Σ_s^- is motivated in part by Theorem 2, which justifies the word "constrained" in "restricted constrained Delaunay triangulation."

The following theorem shows that the sites whose extended Voronoi cells lie in part on an extrusion Σ_s^+ must lie in a ball (of modest radius) centered on the midpoint of the segment s. This helps us to efficiently compute the restricted CDT, because the portion of $\mathrm{Vor}|_{\widetilde{\Sigma}} V$ that lies on Σ_s^+ or Σ_s^- depends only on sites near s.

▶ **Theorem 3** (Possession Theorem). *Let $s \in S$ be a segment with endpoints $p, q \in V$ such that $d(p,q) \le \rho\,\mathrm{lfs}(p)$ for $\rho \le 0.47$. Let c be the midpoint of s. Let $v \in V$ be a site whose extended Voronoi cell $\mathrm{Vor}|_{\widetilde{\Sigma}} v$ contains a point $x \in \Sigma_s^+$ or $x \in \Sigma_s^-$. Then v lies in the ball $B(c, \lambda\,\mathrm{lfs}(p))$ with center c and radius $\lambda\,\mathrm{lfs}(p)$, where*

$$\lambda = \sqrt{1-2\rho}\left(1 - \sqrt{1 - \frac{\rho^2}{4\,(1-2\rho)}}\right) + \sqrt{(2-4\rho)\left(1 - \sqrt{1 - \frac{\rho^2}{4\,(1-2\rho)}}\right)}.$$

For the limiting case $\rho = 0.47$, $\lambda \doteq 0.4694$; $B(c, \lambda\,\mathrm{lfs}(p))$ has almost twice the radius of s.

4 Extended Voronoi cell boundaries

There are only two phenomena that can determine the boundary of an extended Voronoi cell. (1) Portions of a cell's boundary may be determined by hyperplanes, each hyperplane being equidistant from two sites. For example, a point on the boundaries of two cells $\mathrm{Vor}|_{\widetilde{\Sigma}} v$

and $\mathrm{Vor}|_{\widetilde{\Sigma}} w$ might lie on the hyperplane that orthogonally bisects the line segment vw. (2) Portions of a cell's boundary may be determined by a shadow cast by a portal. For example, consider a point $x \in \mathrm{Vor}|_{\widetilde{\Sigma}} v$ such that the line segment xv intersects the boundary of a portal P_s. Portal boundaries do not block visibility; hence the set of points on $\widetilde{\Sigma}$ visible from v is closed. But an infinitesimal perturbation of x might cause x to be no longer visible from v. If x is in the principal branch, this may happen because the perturbed xv intersects the relative interior of P_s; if x is in a secondary branch, it may happen because the perturbed xv no longer intersects P_s. We say that a portal *casts a shadow at* x if $x \in \mathrm{Vor}|_{\widetilde{\Sigma}} v$ lies on the boundary of the points on $\widetilde{\Sigma}$ visible from v. A Voronoi cell $\mathrm{Vor}|_{\widetilde{\Sigma}} w$ is not necessarily closed, because its boundary might contain a shadow point $x \in \mathrm{Vor}|_{\widetilde{\Sigma}} v$ such that $\widehat{d}(v,x) < \widehat{d}(w,x)$.

The following theorem states sampling conditions under which the second phenomenon cannot happen, so the boundaries of all the extended Voronoi cells are determined solely by bisecting hyperplanes, all the extended Voronoi cells are closed point sets, and every point on $\widetilde{\Sigma}$ is in an extended Voronoi cell. For a site $v \in V$, v's *principal Voronoi cell* $\mathrm{Vor}|_{\overline{\Sigma}_S} v = \overline{\Sigma}_S \cap \mathrm{Vor}|_{\widetilde{\Sigma}} v$ is the subset of v's extended Voronoi cell in the principal branch, including the portal curves but excluding the remainder of the extrusions.

▶ **Theorem 4** (Shadow Theorem). *Let S be a set of segments (with endpoints in V) such that for every segment $s = pq \in S$, $d(p,q) \le 0.47\,\mathrm{lfs}(p)$. Suppose that for every site $v \in V$ and every point x in the principal Voronoi cell $\mathrm{Vor}|_{\overline{\Sigma}_S} v$, $d(v,x) \le \max\{\mathrm{lfs}(v), \mathrm{lfs}(x)\}$. (This last condition holds if V is a constrained ϵ-sample, as defined in Section 5, for some $\epsilon \le 1$.)*

Then for every site $v \in V$ and every point x in the extended Voronoi cell $\mathrm{Vor}|_{\widetilde{\Sigma}} v$, the relative interior of the line segment xv does not intersect the boundary of a portal.

▶ **Corollary 5.** *Under the conditions of Theorem 4, every extended Voronoi cell is a closed point set (closed with respect to the topological space $\widetilde{\Sigma}$ or $\widetilde{X}$).*

▶ **Corollary 6.** *Under the conditions of Theorem 4, for every site $v \in V$ and every point x on the boundary of $\mathrm{Vor}|_{\widetilde{\Sigma}} v$, there is a site $w \in V \setminus \{v\}$ such that $x \in \mathrm{Vor}|_{\widetilde{\Sigma}} w$ and $d(v,x) = d(w,x)$.*

▶ **Corollary 7.** *Under the conditions of Theorem 4, if every connected component of Σ contains at least one site in V, then every point on $\widetilde{\Sigma}$ is in at least one extended Voronoi cell.*

The proofs of the Shadow Theorem and its corollaries appear in the full-length article [17].

5 Topological guarantees

Here we introduce conditions under which a restricted CDT is homeomorphic to the surface Σ, with a view toward applications in guaranteed-quality surface mesh generation. The *nearest point map* ν (nu) maps a point $x \in \mathbb{R}^3 \setminus M$ to the point $\nu(x)$ nearest x on Σ. We show that the nearest point map (with its domain restricted to $|\mathrm{Del}|_{\widetilde{\Sigma}} V|$) is a homeomorphism from the underlying space of the restricted CDT $\mathrm{Del}|_{\widetilde{\Sigma}} V$ to the surface Σ.

Our proof has three conditions: *a segment length condition*, that each segment $s \in S$ with endpoints p and q satisfies $d(p,q) \le 0.3647\,\mathrm{lfs}(p)$; a *sampling condition* requiring the sites V to be sufficiently dense; and an *encroachment condition* that prevents vertices in V from being too close to a segment, to prevent the possibility of triangles with excessively large circumcircles. We build on a long line of theoretical work for proving that certain triangulations are topologically correct approximations of surfaces [1, 2, 4, 5, 6, 8, 12, 14, 16], developed to support provably good algorithms for surface reconstruction and mesh generation. Many RDT-based surface mesh generation algorithms enforce a sampling condition by inserting new vertices on Σ [6, 7, 11, 12, 22], and some support constraining segments by inserting additional

vertices that subdivide segments until the RDT naturally respects them [9, 10, 12, 15, 23]. Our three conditions can likewise be enforced by inserting new vertices, but restricted CDTs will often reduce the number of new vertices needed on the segments.

To understand the sampling condition, consider a surface $\Sigma \subset \mathbb{R}^3$ without boundary, a set of segments S with their endpoints on Σ, and a set of portal curves $Z = \{\zeta_s : s \in S\}$. Recall the principal surface $\overline{\Sigma}_S$, defined in Section 2 to be the completion of $\Sigma - \bigcup_{s \in S} \operatorname{relint} \zeta_s$. We say that a finite vertex set $V \subset \Sigma$ is a *constrained ϵ-sample of* (Σ, S, Z) if V contains every endpoint of every $s \in S$ and for every point $x \in \overline{\Sigma}_S$, there is a site $v \in V$ such that $\widehat{d}(x, v) \leq \epsilon \operatorname{lfs}(x)$. That is, the ball with center x and radius $\epsilon \operatorname{lfs}(x)$ contains at least one site visible from x. Here, visibility and $\widehat{d}$ are as defined in Section 2.2; they are what differentiates a constrained ϵ-sample from a standard ϵ-sample. (If S is empty, the two are the same.) Our homeomorphism proof requires that V be a constrained 0.3202-sample of (Σ, S, Z).

The encroachment condition applies only to restricted Delaunay triangles whose dual faces intersect an extrusion, as in Figure 6. (A triangle's dual face is usually a single point, called an extended Voronoi vertex, but our homeomorphism proof does not depend on it.) Let τ be such a triangle. The *circumradius* r of τ is the radius of the unique circle that passes through τ's three vertices. Let w be the vertex of τ at τ's largest plane angle. We require that $r \leq 0.3606 \operatorname{lfs}(w)$. The purpose of this restriction is to prevent the existence of "inverted" triangles in $\operatorname{Del}|_{\widetilde{\Sigma}} V$, which create *foldovers*, points where the nearest point map ν from $|\operatorname{Del}|_{\widetilde{\Sigma}} V|$ to Σ is not locally injective (hence ν is not a homeomorphism).

To put the encroachment condition into perspective, suppose that Σ is a sphere and consider a segment $s \in S$ having the maximum safe length of 0.3647 times the sphere's radius. A triangle τ whose dual vertex lies on Σ_s^+ can exceed the safe radius of 0.3606 times the sphere's radius only if τ has an angle greater than 149.62°. If s is shorter, this angle is larger: in the limit as the segment lengths approach zero (or the radius of Σ approaches infinity), the encroachment condition falls away and restricted CDTs on Σ behave like CDTs in the plane. By contrast, in standard approaches using an RDT that conforms to the segments, no triangle with edge s can have an angle opposite s greater than 90°.

The sampling and encroachment conditions both rule out triangles with circumradii that are excessively large relative to the local feature size. A large circumradius implies either that the triangle is large, or that it has a large plane angle (close to 180°). Imposing these conditions is consistent with a mesh generator's goal of producing only well-shaped triangles, so our conditions are not onerous. Nevertheless, there are other applications such as surface reconstruction where the encroachment condition is not a natural condition. The restricted CDT may nevertheless still be useful in that context; see the Conclusions for speculations.

Our main topological result is the next theorem. Unfortunately, the proof is over twenty pages long; see the full-length article [17]. To keep the proof from being even longer, we assume that no point on $\widetilde{\Sigma}$ is equidistant from four visible vertices in V, which implies that no point is in more than three cells. (This assumption is not actually necessary.)

▶ **Theorem 8** (Homeomorphism). *Let V be a constrained ϵ-sample of (Σ, S, Z) for some $\epsilon \leq 0.3202$. Suppose that for every segment $pq \in S$, $d(p, q) \leq 0.3647 \operatorname{lfs}(p)$. Suppose that no portal curve in Z has a relative interior that intersects another portal curve in Z or a site in V. Suppose that no point on $\widetilde{\Sigma}$ is equidistant from four visible vertices in V. Suppose that for every restricted Delaunay triangle τ whose dual extended Voronoi face intersects an extrusion, τ satisfies $r \leq 0.3606 \operatorname{lfs}(w)$, where r is τ's circumradius and w is the vertex of τ at τ's largest plane angle. Then the nearest point map $\nu : |\operatorname{Del}|_{\widetilde{\Sigma}} V| \to \Sigma$ is a homeomorphism.*

The proof is related to proofs by Amenta et al. [3] and Boissonnat and Ghosh [5] that also use the nearest point map. We sketch a few ideas. To make every extended Voronoi cell become a topological disk and to clarify the duality between the extended Voronoi diagram and the restricted CDT, it is convenient to define a compact 2-manifold without boundary $\mathring{\widetilde{\Sigma}}$, obtained from $\widetilde{\Sigma}$ by gluing each pair Σ_s^+ and Σ_s^- together along their boundaries as illustrated in Figure 7. For each segment $s = pq$, we topologically identify the ray p_s^+ with the ray p_s^-, and q_s^+ with q_s^-. (Theorem 2 shows that p_s^+ and p_s^- are subsets of p's extended Voronoi cell, so this gluing does not confuse which points are in which Voronoi cell.) We create a single point s^∞ "at infinity" (one such point per segment s) at the end of every ray x_s^+ and x_s^- for all $x \in \zeta_s$, thereby making $\mathring{\widetilde{\Sigma}}$ compact. Thus the hole created in $\overline{\Sigma}_S$ by cutting a slit at ζ_s is filled in $\mathring{\widetilde{\Sigma}}$ with a topological disk $\Sigma_s^+ \cup \Sigma_s^- \cup s^\infty$, as illustrated. Clearly, $\mathring{\widetilde{\Sigma}}$ is homeomorphic to Σ. (Note that unlike $\widetilde{\Sigma}$, $\mathring{\widetilde{\Sigma}}$ is not embedded in $\widetilde{X}$ and has neither coordinates nor distances.)

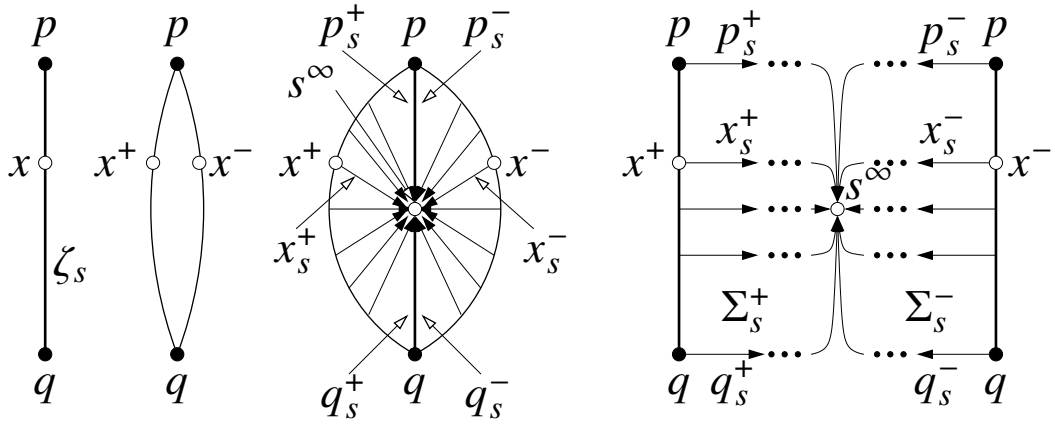


Figure 7 After we remove relint ζ_s from Σ, we glue two extrusions Σ_s^+ and Σ_s^- in the hole to create $\widetilde{\Sigma}$. Additional gluing can transform $\widetilde{\Sigma}$ into a compact 2-manifold without boundary $\mathring{\widetilde{\Sigma}}$, restoring the topology of Σ, by gluing the ray p_s^+ to p_s^-, gluing q_s^+ to q_s^-, and filling the hole with a point s^∞.

If V is a constrained 0.44-sample and S satisfies the segment length condition, we show that every extended Voronoi cell on $\mathring{\widetilde{\Sigma}}$ is homeomorphic to a closed disk. With the assumption that no point is in more than three cells, this implies that the adjacency graph of $\mathrm{Vor}|_{\widetilde{\Sigma}} V$, which is the graph of $\mathrm{Del}|_{\widetilde{\Sigma}} V$, can be drawn on $\mathring{\widetilde{\Sigma}}$ (and therefore on $\widetilde{\Sigma}$) with no crossings.

We call an extended Voronoi vertex a *principal vertex* if it lies in the principal branch (on $\overline{\Sigma}_S$), or a *secondary vertex* if it lies on an extrusion but not on a portal curve. We show that if V is a constrained 0.3202-sample, each principal vertex dualizes to a triangle whose circumradius is not large (relative to the local feature size). The encroachment condition implies that each secondary vertex dualizes to a triangle whose circumradius is not large.

The bounds on circumradii allow us to prove that the nearest point map restricted to any single restricted Delaunay triangle is a homeomorphism. Moreover, there is a sense in which the map preserves orientation: for any extended Voronoi vertex u whose dual extended Delaunay triangle is $\tau = \triangle pp'p''$, the sites p, p', and p'' are in counterclockwise order around $\nu(\tau)$ if and only if the cells $\mathrm{Vor}|_{\widetilde{\Sigma}} p$, $\mathrm{Vor}|_{\widetilde{\Sigma}} p'$, and $\mathrm{Vor}|_{\widetilde{\Sigma}} p''$ adjoin u in counterclockwise order around u (as seen from outside Σ). From this, we argue that along each of its edges, each restricted Delaunay triangle adjoins another restricted Delaunay triangle with a consistent orientation, and therefore the triangles must cover the whole surface Σ – that is, the nearest point map is a surjection from $|\mathrm{Del}|_{\widetilde{\Sigma}} V|$ to Σ. As the boundary of a Voronoi cell $\mathrm{Vor}|_{\widetilde{\Sigma}} v$ is a simple loop, the restricted Delaunay triangles adjoining v form a fan of triangles around v whose union is a topological disk. From these facts we prove that the nearest point map is an injection from $|\mathrm{Del}|_{\widetilde{\Sigma}} V|$ to Σ (there are no foldovers) and therefore a homeomorphism.

6 Conclusions

The restricted constrained Delaunay triangulation is a rigorous generalization of the constrained Delaunay triangulation to surfaces. Under suitable conditions on the vertex density and the segment lengths, the restricted CDT is homeomorphic to Σ and contains every constraining segment. We believe that the restricted CDT will become a useful tool for enforcing specified boundaries in guaranteed-quality algorithms for surface meshing. But first and foremost, we think the existence of restricted CDTs is a beautiful mathematical fact.

Several algorithms suggest themselves for computing the restricted CDT. The classical gift-wrapping algorithm [12, Section 3.11] [25] can be adapted. Another approach, likely faster in practice, is to start with the RDT and then incrementally insert the segments one by one [27]. It is an open problem to design an algorithm that runs in $O(|V|^2)$ time or better.

Another open problem is to design a guaranteed-quality algorithm that uses the restricted CDT to mesh surfaces with prescribed boundaries. The algorithm must generate new vertices on Σ with the goal of enforcing the sampling and encroachment conditions, in addition to the customary goal of constructing high-quality triangles. As we have said, we believe that restricted CDTs will require fewer triangles and vertices than algorithms based on RDTs.

Although our encroachment condition is reasonable in a surface mesh generator, it is undesirable in some applications such as surface reconstruction. Unfortunately, without this condition, we cannot prove a homeomorphism because the nearest point map is not necessarily injective. Figure 8 illustrates the problem. Suppose that we place two segments s and s' close together and we place a vertex r very close to the midpoint of $\zeta_{s'}$ (violating the encroachment condition), as shown in Figure 8. Consider the triangle formed by r and s', and its dual 3D Voronoi edge e; e can be arbitrarily close to perpendicular to $P_{s'}$. Then e may enter both portals P_s and $P_{s'}$ and generate two extended Voronoi vertices (illustrated in red) where it intersects $\Sigma^+_{s'}$ (which is desirable) and Σ^+_s (which is not). This circumstance is possible because the segments are close together, the portals are tilted toward each other, and Σ^+_s is extruded infinitely far. Increasing the sampling density does not fix this problem.

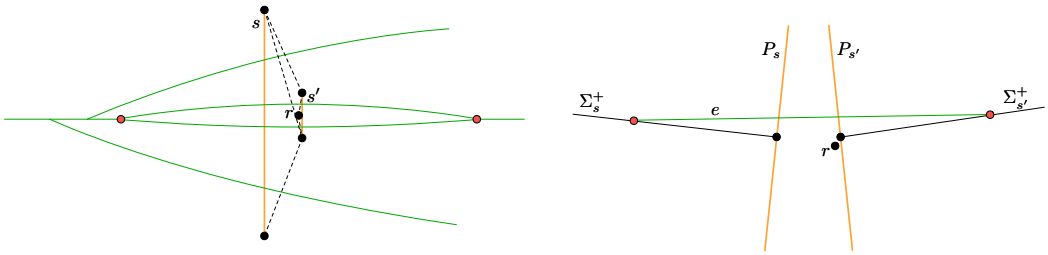


Figure 8 The left figure shows a top view of a Voronoi diagram drawn on $\widetilde{\Sigma}$; the right figure shows a side view. The segments s and s' are placed close together on Σ and their portals P_s and $P_{s'}$ are tilted toward each other in the side view. If r is arbitrarily close to $P_{s'}$, the Voronoi edge e dual to $\triangle rs'$ is tilted nearly tangent to the surface and can leave $P_{s'}$, enter P_s, and intersect Σ^+_s (perhaps far down the extrusion). The dual triangulation contains a dangling triangle $\triangle rs'$.

If we drop the encroachment condition (but retain the other two conditions), we conjecture that the restricted Delaunay triangles still form a watertight enclosure such that the nearest point map is a surjection from the restricted CDT to Σ. However, it is not necessarily injective; there may be foldovers where sites brush up against segments. There may also be "dangling" triangles, connected to the remainder of the triangulation by only a single edge; an example is the triangle formed by r and s' in Figure 8. Such triangles are easily pruned.

References

1 Nina Amenta and Marshall Bern. Surface Reconstruction by Voronoi Filtering. *Discrete & Computational Geometry*, 22(4):481–504, June 1999.

2 Nina Amenta, Marshall W. Bern, and David Eppstein. The Crust and the β-Skeleton: Combinatorial Curve Reconstruction. *Graphical Models and Image Processing*, 60(2):125–135, March 1998.

3 Nina Amenta, Sunghee Choi, Tamal K. Dey, and Naveen Leekha. A Simple Algorithm for Homeomorphic Surface Reconstruction. *International Journal of Computational Geometry and Applications*, 12(1–2):125–141, 2002.

4 Jean-Daniel Boissonnat, Frédéric Chazal, and Mariette Yvinec. *Geometric and Topological Inference*, volume 57. Cambridge University Press, September 2018.

5 Jean-Daniel Boissonnat and Arijit Ghosh. Manifold Reconstruction Using Tangential Delaunay Complexes. *Discrete & Computational Geometry*, 51(1):221–267, 2014.

6 Jean-Daniel Boissonnat and Steve Oudot. Provably Good Surface Sampling and Approximation. In *Symposium on Geometry Processing*, pages 9–18. Eurographics Association, June 2003.

7 Jean-Daniel Boissonnat and Steve Oudot. Provably Good Sampling and Meshing of Surfaces. *Graphical Models*, 67(5):405–451, September 2005.

8 Ho-Lun Cheng, Tamal K. Dey, Herbert Edelsbrunner, and John Sullivan. Dynamic Skin Triangulation. *Discrete & Computational Geometry*, 25(4):525–568, December 2001.

9 Siu-Wing Cheng, Tamal K. Dey, and Joshua A. Levine. A Practical Delaunay Meshing Algorithm for a Large Class of Domains. In *Proceedings of the 16th International Meshing Roundtable*, pages 477–494, Seattle, Washington, October 2007.

10 Siu-Wing Cheng, Tamal K. Dey, and Edgar A. Ramos. Delaunay Refinement for Piecewise Smooth Complexes. *Discrete & Computational Geometry*, 43(1):121–166, 2010.

11 Siu-Wing Cheng, Tamal K. Dey, Edgar A. Ramos, and Tathagata Ray. Sampling and Meshing a Surface with Guaranteed Topology and Geometry. *SIAM Journal on Computing*, 37(4):1199–1227, 2007.

12 Siu-Wing Cheng, Tamal Krishna Dey, and Jonathan Richard Shewchuk. *Delaunay Mesh Generation*. CRC Press, Boca Raton, Florida, 2012.

13 L. Paul Chew. Constrained Delaunay Triangulations. *Algorithmica*, 4(1):97–108, 1989.

14 Tamal K. Dey. *Curve and Surface Reconstruction: Algorithms with Mathematical Analysis*. Cambridge University Press, New York, 2007.

15 Tamal K. Dey and Joshua A. Levine. Delaunay Meshing of Piecewise Smooth Complexes without Expensive Predicates. *Algorithms*, 2(4):1327–1349, 2009.

16 Herbert Edelsbrunner and Nimish R. Shah. Triangulating Topological Spaces. *International Journal of Computational Geometry and Applications*, 7(4):365–378, 1997.

17 Marc Khoury and Jonathan Richard Shewchuk. Restricted Constrained Delaunay Triangulations. The complete version of this paper, March 2021. URL: `https://people.eecs.berkeley.edu/~jrs/papers/rcdt.pdf`.

18 Ron Kimmel and James A. Sethian. Computing Geodesic Paths on Manifolds. *Proceedings of the National Academy of Sciences*, 95(15):8431–8435, July 1998.

19 Der-Tsai Lee and Arthur K. Lin. Generalized Delaunay Triangulations for Planar Graphs. *Discrete & Computational Geometry*, 1:201–217, 1986.

20 Takashi Maekawa. Computation of Shortest Paths on Free-Form Parametric Surfaces. *Journal of Mechanical Design*, 118(4):499–508, December 1996.

21 Manish Mandad, David Cohen-Steiner, Leif Kobbelt, Pierre Alliez, and Mathieu Desbrun. Variance-Minimizing Transport Plans for Inter-Surface Mapping. *ACM Transactions on Graphics*, 36(4), 2017.

22 Steve Oudot, Laurent Rineau, and Mariette Yvinec. Meshing Volumes Bounded by Smooth Surfaces. In *Proceedings of the 14th International Meshing Roundtable*, pages 203–219, San Diego, California, September 2005. Springer.

23 Laurent Rineau and Mariette Yvinec. Meshing 3D Domains Bounded by Piecewise Smooth Surfaces. In *Proceedings of the 16th International Meshing Roundtable*, pages 443–460, Seattle, Washington, 2007. Springer.

24 Erich Schönhardt. Über die Zerlegung von Dreieckspolyedern in Tetraeder. *Mathematische Annalen*, 98:309–312, 1928.

25 Raimund Seidel. Constrained Delaunay Triangulations and Voronoi Diagrams with Obstacles. In H. S. Poingratz and W. Schinnerl, editors, *1978–1988 Ten Years IIG*, pages 178–191. Institute for Information Processing, Graz University of Technology, 1988.

26 Jonathan Richard Shewchuk. Delaunay Refinement Algorithms for Triangular Mesh Generation. *Computational Geometry: Theory and Applications*, 22(1–3):21–74, 2002.

27 Jonathan Richard Shewchuk and Brielin C. Brown. Fast Segment Insertion and Incremental Construction of Constrained Delaunay Triangulations. *Computational Geometry: Theory and Applications*, 48(8):554–574, 2015.

Near Neighbor Search via Efficient Average Distortion Embeddings

Deepanshu Kush
University of Toronto, Canada

Aleksandar Nikolov
University of Toronto, Canada

Haohua Tang
University of Toronto, Canada

Abstract

A recent series of papers by Andoni, Naor, Nikolov, Razenshteyn, and Waingarten (STOC 2018, FOCS 2018) has given approximate near neighbour search (NNS) data structures for a wide class of distance metrics, including all norms. In particular, these data structures achieve approximation on the order of p for ℓ_p^d norms with space complexity nearly linear in the dataset size n and polynomial in the dimension d, and query time sub-linear in n and polynomial in d. The main shortcoming is the *exponential in d pre-processing time* required for their construction.

In this paper, we describe a more direct framework for constructing NNS data structures for general norms. More specifically, we show via an algorithmic reduction that an efficient NNS data structure for a metric $\mathcal{M}$ is implied by an efficient average distortion embedding of $\mathcal{M}$ into ℓ_1 or the Euclidean space. In particular, the resulting data structures require only *polynomial pre-processing time*, as long as the embedding can be computed in polynomial time.

As a concrete instantiation of this framework, we give an NNS data structure for ℓ_p with *efficient pre-processing* that matches the approximation factor, space and query complexity of the aforementioned data structure of Andoni et al. On the way, we resolve a question of Naor (Analysis and Geometry in Metric Spaces, 2014) and provide an explicit, efficiently computable embedding of ℓ_p, for $p \geq 1$, into ℓ_1 with average distortion on the order of p. Furthermore, we also give data structures for Schatten-p spaces with improved space and query complexity, albeit still requiring exponential pre-processing when $p \geq 2$. We expect our approach to pave the way for constructing efficient NNS data structures for all norms.

2012 ACM Subject Classification Theory of computation → Nearest neighbor algorithms

Keywords and phrases Nearest neighbor search, metric space embeddings, average distortion embeddings, locality-sensitive hashing

Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.50

Related Version *Full Version*: `https://arxiv.org/abs/2105.04712`

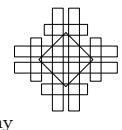
Funding *Aleksandar Nikolov*: Supported by an NSERC Discovery Grant (RGPIN-2016-06333), and a Tier II Canada Research Chair.
Haohua Tang: Supported by a Tier II Canada Research Chair.

Acknowledgements We want to thank Sushant Sachdeva for a fruitful discussion.

1 Introduction

In the nearest neighbor problem, a fundamental problem in computational geometry, we are given an n-point subset P of a metric space $\mathcal{M}$ with a distance function $\mathrm{d}_{\mathcal{M}}$, and our goal is to preprocess P into a data structure that, given a query point $q \in \mathcal{M}$, finds a point $x \in P$ minimizing $\mathrm{d}_{\mathcal{M}}(x, q)$. The main parameters of a nearest neighbor search data structure are

- the *pre-processing* time required to construct the data structure given P;
- the *space* taken up by the data structure, in words of memory;
- the *query time* required to answer a nearest neighbor query.




A trivial solution is to store P as a list of points and to answer queries by linear search. Ignoring the time required to compute distances, this solution takes $\Theta(n)$ space, but also requires $\Theta(n)$ query time, which is prohibitively large when we have a large data set and expect to answer many queries. However, in some cases it is possible to use the geometry of the metric to design data structures with much more efficient query procedures and nearly the same space requirements. For instance, Lipton and Tarjan [19] gave a data structure for the nearest neighbor problem in the 2-dimensional Euclidean plane with $O(n \log n)$ pre-processing, $O(n)$ space, and $O(\log n)$ query time. This result has been extended to d-dimensional Euclidean space (see e.g. [21]), and other d-dimensional normed spaces. There is, however, no known nearest neighbor data structure for the d-dimensional Euclidean space that achieves space that is polynomial in n and d, and query time that is polynomial in d, and sublinear in n (i.e., $O(\text{poly}(d) \cdot n^{1-\alpha})$ for some $\alpha > 0$).[1] There is, furthermore, some evidence that no such data structure exists [30].

The nearest neighbor search problem finds a multitude of applications beyond computational geometry, in areas as diverse as databases, computer vision, and machine learning. For example, it is used to find joinable tables in publicly available data [24]; for object recognition [22] and shape matching [10] in computer vision; to solve analogical reasoning tasks [23]; in machine learning, the k-Nearest Neighbors classifier is a common baseline. In these applications, often both the data set size n and the dimension d are large, making query times that are linear in n or exponential in d unacceptable. It makes sense then to relax this problem in the hope of allowing for efficient data structures in the high-dimensional regime. A common relaxation is to allow returning an approximate nearest neighbor to the query point q, i.e., a point $x \in P$ for which $\mathrm{d}_{\mathcal{M}}(x, q) \leq c \min_{y \in P} \mathrm{d}_{\mathcal{M}}(y, q)$ for some approximation factor $c > 1$. A long and fruitful line of work, recently surveyed in [4], has shown that it is possible to construct data structures for this approximate nearest neighbor problem over certain spaces such as the d-dimensional Euclidean or Manhattan distance that use space $O(n^{1+\varepsilon} \cdot \text{poly}(d))$ and support queries in time $O(n^{\varepsilon} \cdot \text{poly}(d))$, for a constant $\varepsilon < 1$ that goes to 0 as the approximation factor c goes to infinity.

Rather than solving the nearest neighbor search problem directly, it is more convenient to fix a scale for the distance, and work with the (c, r)-near neighbor search ((c, r)-NNS) problem, defined below.

▶ **Definition 1.** *In the (c, r)-near neighbor search ((c, r)-NNS) problem, we are given a set of n points P in a metric space $(\mathcal{M}, \mathrm{d}_{\mathcal{M}})$, and are required to build a data structure so that given a query point $q \in \mathcal{M}$ with the guarantee that $\mathrm{d}_{\mathcal{M}}(x^*, q) \leq r$ for some $x^* \in P$, we can use the data structure to output a point $x \in P$ satisfying $\mathrm{d}_{\mathcal{M}}(x, q) \leq cr$ with probability at least $\frac{2}{3}$.*

It was shown by Indyk and Motwani [15] that the approximate nearest neighbor problem can be reduced to solving $\text{poly}(\log n)$ instances of the (c, r)-NNS problem. Therefore, we focus on the latter problem from this point onward.

Most (but not all) efficient data structures for the NNS problem in the high-dimensional regime are based on the idea of locality sensitive hashing (LSH), introduced by Indyk and Motwani [15]. A locality sensitive family of hash functions is a probability distribution $\mathcal{H}$ over random functions $h : \mathcal{M} \to \Omega$ such that pairs of close points are much more likely to be mapped by h to the same value than far points. In particular, pairs of points at distance at

[1] Here and in the rest of the paper, we use the notation $\text{poly}(A)$ to denote the class of polynomials in the expression A.

most r get mapped to the same bucket with probability at least p_1, while pairs of points at distance at least cr get mapped to the same bucket with probability at most p_2, with $p_1 > p_2$. Indyk and Motwani showed that an LSH family implies a data structure for the (c, r)-NNS problem with space $O(n^{1+\rho} \log_{1/p_2}(n))$, and query time $O(n^{\rho} \log_{1/p_2}(n))$, where $\rho = \frac{\log(1/p_1)}{\log(1/p_2)}$. Moreover, they constructed LSH families for the Hamming and Manhattan (i.e., ℓ_1) distance with $\rho \approx \frac{1}{c}$. Subsequent work also showed the existence of LSH families for the Euclidean distance (i.e., ℓ_2), as well as the ℓ_p metric for $1 \leq p \leq 2$, and improved the parameters [2, 12]. The LSH definition above has the property that the distribution $\mathcal{H}$ is independent of the dataset P. Sometimes, however, data structures with better trade-offs can be constructed by allowing $\mathcal{H}$ to depend on properties of P [3, 5, 8].

Until recently, relatively little was known about the NNS problem beyond the ℓ_p spaces for $1 \leq p \leq 2$, and the ℓ_∞^d space,[2] for which Indyk gave an efficient deterministic decision tree data structure with approximation $O(\log \log d)$ [14]. Data structures for other spaces can be constructed by reducing to these special cases via bi-Lipschitz embeddings. I.e., if for some metric $\mathcal{M}$ we can find an efficiently computable injection $f : \mathcal{M} \to \ell_2^d$ such that $\|f(x) - f(y)\|_2 \approx \mathrm{d}_{\mathcal{M}}(x, y)$ for all $x, y \in \mathcal{M}$, then we can use NNS data structures for ℓ_2^d to solve the NNS problem in $\mathcal{M}$. The best approximation factor achievable by this approach depends on the *distortion* of f, which measures how well $\|f(x) - f(y)\|_2$ approximates $\mathrm{d}_{\mathcal{M}}(x, y)$ in the worst case, and is defined as $\|f\|_{\mathrm{Lip}} \cdot \|f^{-1}\|_{\mathrm{Lip}}$, where

$$\|f\|_{\mathrm{Lip}} \coloneqq \sup_{x \neq y, x, y \in \mathcal{M}} \frac{\|f(x) - f(y)\|}{\mathrm{d}_{\mathcal{M}}(x, y)}, \qquad \|f^{-1}\|_{\mathrm{Lip}} \coloneqq \sup_{x \neq y, x, y \in \mathcal{M}} \frac{\mathrm{d}_{\mathcal{M}}(x, y)}{\|f(x) - f(y)\|_2}$$

are the Lipschitz constants of f and its inverse, respectively. Although this approach does yield some non-trivial results (see [4] for a survey), it only produces data structures with approximation $c \geq d^{\frac{1}{2} - \frac{1}{p}}$ even in the special case of ℓ_p^d (with $p > 2$), as this is the best possible distortion achievable by a bi-Lipszhitz embedding into ℓ_2 (see, e.g., [11]). It is natural to ask if the best approximation achievable by an efficient NNS data structure for a metric $\mathcal{M}$ is characterized by the optimal distortion of a bi-Lipschitz embedding into ℓ_2. More generally, a fundamental problem in high-dimensional computational geometry is to *determine the geometric properties of a metric space that allow efficient and accurate NNS data structures.*

A recent line of work showed that the answer to the first question above is negative, and there exist efficient NNS data structures with approximation much better than what is implied by bi-Lipszhitz embeddinds into ℓ_2^d or ℓ_1^d [6, 7]. These papers give data-dependent LSH families for any d-dimensional normed space with approximation factor that is sub-polynomial in the dimension d. A sample theorem is the following.

▶ **Theorem 2** ([6]). *For any $r > 0$, $p \geq 2$ and any $\varepsilon \in (0, 1)$, there is some $c \lesssim \frac{p}{\varepsilon}$ such that the following holds. For any set P of n points in $\mathbb{R}^d$ such that for all $x \in \mathbb{R}^d$ we have $|B_{\ell_p^d}(x, cr) \cap P| \leq \frac{n}{2}$, there exists a probability distribution on axis-aligned boxes S satisfying*

$$\mathbb{P}\left[\frac{n}{4} \leq |S \cap P| \leq \frac{3n}{4}\right] = 1$$
$$\|x - y\|_p \leq r \implies \mathbb{P}[|S \cap \{x, y\}| = 1] \leq \varepsilon.$$

[2] Recall the ℓ_p^d norm on $\mathbb{R}^d$: $\|x\|_p \coloneqq \left(\sum_{i=1}^d |x_i|^p\right)^{1/p}$ for $1 \leq p < \infty$, and $\|x\|_\infty \coloneqq \max_{i=1}^d |x_i|$.

Here, $B_{\ell_p^d}(x, cr) = \{y \in \mathbb{R}^d : \|y-x\|_p \leq cr\}$ is the ℓ_p^d-ball of radius cr centered at x. Moreover, here and in the rest of the paper, the notation $A \lesssim B$ means that there exists an absolute constant C, independent of all other parameters, such that $A \leq CB$.

Theorem 2 gives a form of LSH: we can think of points inside S as being mapped to 1, and points outside mapped to 0. We still have the p_1 condition: close points get mapped to the same value with probability at least $1 - \varepsilon$. Rather than guaranteeing that far points are less likely to be mapped to the same value, we have a data-dependent condition: if the data set contains no dense clusters of points, then most pairs of points are mapped to different values. It is not hard to construct a randomized decision tree using Theorem 2, and [6] showed how to use it to give a data structure for (c, r)-NNS over ℓ_p^d with approximation $c \lesssim \frac{p}{\varepsilon}$, space $O(n^{1+\varepsilon} \cdot \text{poly}(d))$ and query time $O(n^{\varepsilon} \cdot \text{poly}(d))$. Similar results were also proved for the Schatten-p norms, which extend the ℓ_p norms to matrices, and for arbitrary norms (with appropriate access to the norm ball) in [7].

Theorem 2, however, has a significant shortcoming: it does not guarantee that the distribution over axis-aligned boxes can be sampled efficiently given P as input. Indeed, the proof of the theorem in [6], as well as the proofs of other similar results in [6, 7], rely on a duality argument and yield sampling algorithms with running time exponential in the dimension. For this reason, the resulting data structures have pre-processing time that is also exponential in the dimension. These works thus raise an intriguing open problem: *can we sample a distribution such as the one in Theorem 2 in time polynomial in n and d?*

1.1 Our Results on Near Neighbor Search

In this work, we resolve the open problem above (also posed explicitly in [6]), and prove the following theorem.

▶ **Theorem 3.** *Let $\varepsilon \in (0, 1]$, $r > 0$, $p \geq 2$. For some $c \lesssim \frac{p}{\varepsilon}$, there exists a data structure for the (c, r)-NNS problem over n-point sets in ℓ_p^d with*

- *pre-processing time* $\text{poly}(nd)$*;*
- *space* $O(n^{1+\varepsilon} \log(n) \cdot \text{poly}(d))$*;*
- *query time* $O(n^{\varepsilon} \log(n) \cdot \text{poly}(d))$.

We note that the only previous NNS data structure over ℓ_p^d (for $p > 2$) with pre-processing time $\text{poly}(nd)$ could only achieve approximation on the order of 2^p, and used polynomial rather than nearly linear space [28, 9].

We further extend this result to the Schatten-p norms, which are a natural extension of ℓ_p^d to matrices. For a $d \times d$ symmetric real matrix X and $p \in [1, \infty]$, the Schatten-p norm $\|X\|_{C_p}$ of X is defined as the ℓ_p norm of the eigenvalues of X. In addition to their intrinsic interest [16, 17, 18, 27], the Schatten-p spaces are an interesting first step when extending geometric and analytic results from the ℓ_p spaces to more general norms: while Schatten-p shares many properties with ℓ_p, extending proofs and algorithms from ℓ_p to Schatten-p requires finding coordinate-free and often, more natural arguments. Here, we partially succeed in extending Theorem 3 to Schatten-p spaces, and show the following two theorems.

▶ **Theorem 4.** *Let $\varepsilon \in (0, 1]$, $r > 0$, $1 \leq p \leq 2$. For some $c \lesssim \frac{1}{\varepsilon^{2/p}}$, there exists a data structure for the (cr, r)-NNS problem over n-point sets of $d \times d$ symmetric matrices with respect to the Schatten-p norm with*

- *pre-processing time* $\text{poly}(nd)$*;*
- *space* $O(n^{1+\varepsilon} \log(n) \cdot \text{poly}(d))$*;*
- *query time* $O(n^{\varepsilon} \log(n) \cdot \text{poly}(d))$.

The only previously known NNS data structures for Schatten-p with constant approximation and $\mathrm{poly}(nd)$ pre-processing time have polynomial rather than nearly linear space complexity. While not explicitly described there, such data structures follow from the techniques in [28, 9], in combination with the results in [29].

▶ **Theorem 5.** *Let $\varepsilon \in (0,1]$, $r > 0$, $p \geq 2$. For some $c \lesssim \frac{1}{\varepsilon}$, there exists a data structure for the (cr, r)-NNS problem over n-point sets of $d \times d$ symmetric matrices with respect to the Schatten-p norm with*

- *pre-processing time* $\mathrm{poly}(n) \cdot 2^{\mathrm{poly}(d)}$*;*
- *space* $O(n^{1+\varepsilon} \log(n) \cdot \mathrm{poly}(d))$*;*
- *query time* $O(n^{\varepsilon} \log(n) \cdot \mathrm{poly}(d))$*.*

In this theorem, the pre-processing time is exponential in the dimension as it is, as well, in [6]. Nevertheless, the data structure in Theorem 5 has the benefit that it has query time polynomial in d, rather than polynomial in d^p, as in the data structure in [6].

1.2 Techniques and Results on Average Distortion Embeddings

To prove Theorems 3, 4, and 5, we develop a new approach for proving partitioning statements such as Theorem 2 that relies on the notion of embeddings with average distortion, defined below (this definition is taken from [26]).

▶ **Definition 6.** *Given two metric spaces $(\mathcal{M}, \mathrm{d}_{\mathcal{M}})$ and $(\mathcal{N}, \mathrm{d}_{\mathcal{N}})$, and an n-point set $P \subseteq \mathcal{M}$, we say a function $f : \mathcal{M} \to \mathcal{N}$ is an embedding of $\mathcal{M}$ into $\mathcal{N}$ with q-average distortion D (with respect to P) if*

$$\sum_{x \in P} \sum_{y \in P} \mathrm{d}_{\mathcal{N}}(f(x), f(y))^q \geq \frac{\|f\|_{\mathrm{Lip}}^q}{D^q} \sum_{x \in P}^{n} \sum_{y \in P}^{n} \mathrm{d}_{\mathcal{M}}(x, y)^q.$$

As before, $\|f\|_{\mathrm{Lip}}$ is the Lipschitz constant of f, i.e., $\|f\|_{\mathrm{Lip}} = \sup_{x \neq y, x, y \in \mathcal{M}} \frac{\mathrm{d}_{\mathcal{N}}(f(x), f(y))}{\mathrm{d}_{\mathcal{M}}(x, y)}$. When the embedding has 1-average distortion D, we simply say it has average distortion D. If for every integer n and any n point set in $\mathcal{M}$, there exists an embedding into $\mathcal{N}$ with average distortion D, then we say that $\mathcal{M}$ embeds into $\mathcal{N}$ with average distortion D.

Somewhat informally, we show that if a metric space $\mathcal{M}$ embeds into ℓ_1^d with average distortion D via an embedding f that can be efficiently computed from P, and efficiently evaluated, then $\mathcal{M}$ supports NNS data structures with approximation $c \lesssim \frac{D \log D}{\varepsilon}$, space $O(n^{1+\varepsilon})$, query time $O(n^{\varepsilon})$, and efficient pre-processing. (Precise statements follow from Lemmas 14 and 19, and Theorem 20 below.) This connection strengthens the reductions via bi-Lipschitz embeddings mentioned above. Moreover, this connection between NNS and average distortion embeddings is closely related to the connection between NNS and the cutting modulus from prior work [6]. In particular, bounds on the cutting modulus in [6] were proved by utilizing comparison inequalities between non-linear spectral gaps, in the sense of [25]. Such comparison inequalities were shown in [25] to be equivalent to the existence of average distortion embeddings. However, the connection between the cutting modulus and NNS data structures in [6] is not algorithmic: it involves a non-algorithmic duality argument that only yields data structures with exponential pre-processing, even when the cutting modulus bound is witnessed by an efficiently computable average distortion embedding. By contrast, our new connection between average distortion embeddings and NNS data structures is an efficient reduction: if the embedding is computationally efficient, so is the data structure, including the pre-processing.

To reduce constructing efficient NNS data structures to finding efficient average distortion embeddings, we first formalize the type of data dependent LSH family implicit in Theorem 2 and in other similar results. As mentioned above, these data-dependent LSH families relax the p_2 requirement of the standard LSH definition, by requiring that it holds empirically for the input point set P. I.e., we require that each hash function in the family maps at least $1 - p_2$ fraction of the pairs of points in P to different values (see Definition 9 for the precise requirement). As noted above, such a data-dependent LSH family is still sufficient to design an NNS data structure with similar running time and space guarantees as given by standard LSH (Lemma 14). Moreover, it is also not hard to construct a data-dependent LSH family using a standard LSH family when the point set P is dispersed, i.e., when no ball of radius cr contains more than half of P (Lemma 15).

So far these results just give a different perspective on standard NNS data structures using LSH. The benefit of using data-dependent LSH, however, is that the data-dependent requirement allows using a larger class of embeddings in reductions. While the existence of a standard LSH family for, e.g., ℓ_1^d, is inherited by all metrics that have a bi-Lipschitz embedding into ℓ_1^d with small distortion,[3] the existence of a data-dependent LSH family for ℓ_1^d is inherited by metrics $\mathcal{M}$ that have, for any dispersed point set $P \subseteq \mathcal{M}$, an embedding f into ℓ_1^d which (1) does not expand distances too much, and (2) does not map a dispersed point set P into a point set that is not dispersed. We formally define this class of embeddings, which we call embeddings with weak average distortion, in Definition 6 below. To finish the connection between NNS and average distortion embeddings, we prove that the existence of (computationally efficient) average distortion embeddings implies the existence of (computationally efficient) weak average distortion embeddings. The proof of this fact uses ideas previously used to relate embeddings with p- and q-average distortion (see Section 5.1 in [26]).

Finally, in order to utilize this general connection between average distortion embeddings and NNS data structures, we need to construct explicit, efficiently computable average distortion embeddings into ℓ_1^d or ℓ_2^d. Naor has shown that the existence of average distortion embeddings of a metric space $\mathcal{M}$ into ℓ_2 is equivalent to proving a certain inequality between non-linear spectral gaps, and, using this equivalence, he showed that when $p \geq 2$ ℓ_p^d embeds into ℓ_2 with average distortion $D \lesssim p$ [25, 26]. The connection between average distortion embeddings and spectral gap inequalities, however, uses a duality argument, and does not provide explicit, efficiently computable embeddings. In fact, an explicit construction of an embedding of ℓ_p^d into ℓ_2 is given as an open problem in [25]. Here we resolve (a variant of) this open problem. In the theorem below, the functions $M_{p,1}, \widetilde{M}_{p,1} : \ell_p^d \to \ell_1^d$ are defined by

$$M_{p,1}(x) = (\text{sign}(x_i)|x_i|^p)_{i=1}^d \qquad \widetilde{M}_{p,1}(x) = \|x\|_p M_{p,1}\left(\frac{x}{\|x\|_p}\right) = \|x\|_p^{1-p} M_{p,1}(x),$$

with $\widetilde{M}_{p,1}(0) = 0$.

▶ **Theorem 7.** *For any $p \geq 1$, and any n-point set P in $\mathbb{R}^d$, for $t \in \mathbb{R}^d$ so that*

$$\forall i \in [d]: \quad |\{x \in P : x_i < t_i\}| = |\{x \in P : x_i > t_i\}|,$$

the map $g : \ell_p^d \to \ell_1^d$ defined by $g(x) = \widetilde{M}_{p,1}(x - t)$ has average distortion $D \lesssim p$.

We note that Naor's open problem was defined for embeddings into ℓ_2 and 2-average distortion. Our techniques can be extended to give similar results for such embeddings as well.

[3] In fact a weaker notion of randomized embedding suffices [4].

Above, $M_{p,1}$ is the classical Mazur map from ℓ_p^d to ℓ_1^d. This map sends the unit sphere in ℓ_p^d to the unit sphere in ℓ_1^d, and its restriction to the sphere has Lipschitz constant bounded by p up to absolute constants. The Mazur map was previously used by Matoušek to prove bounds on non-linear spectral gaps in ℓ_p^d [20]. As mentioned before, such bounds are closely related to the existence of average distortion embeddings, via Naor's duality argument in [25]. The Mazur map itself, however, cannot be used directly to give an average distortion embedding, since its Lipschitz constant is unbounded over all of ℓ_p^d. Our main technical contribution in the proof of Thereom 7 is the observation that the rescaled Mazur map $\widetilde{M}_{p,1}$ has Lipschitz constant $\lesssim p$ everywhere, and that the machinery of Matoušek's spectral gap argument can be then used to prove a bound on the average distortion directly, without going through a duality argument.

Our technique for constructing explicit average distortion embeddings in fact extends to every pair of normed spaces X and Y for which we have a Hölder-continuous homeomorphism f between the unit spheres of X and Y. We can then show that this homeomorphism can be extended to a function $\widetilde{f}$ which is Hölder-continuous on all of X, and that there is a shift $t \in X$ so that the map $g : X \to Y$ defined by $g(x) = \widetilde{f}(x - t)$ is an average distortion embedding of (a snowflake of) X into Y. One can then use a variety of known homemorphisms between spheres and construct reasonably explicit average distortion embeddings. We do so for the Schatten-p spaces, and one can also use the homeomorphism between finite dimensional normed spaces in [7] to give results for general normed spaces, too. Except for some special cases like ℓ_p^d and Schatten-p for $1 \leq p \leq 2$, however, one aspect of these embeddings is still not fully explicit, and in particular, not computationally efficient. Namely, the argument showing that there exists a good shift t, which was first given in [6], uses the theory of topological degree that is also used in textbook proofs of Brouwer's fixed point theorem, and does not suggest an efficient algorithm for computing t. We leave finding such an algorithm, even for the case of Schatten-p norms with $p \geq 2$, as an open problem.

2 Weak Average Distortion Embeddings Imply NNS

We are going to assume that the metric spaces $(\mathcal{M}, \mathrm{d}_{\mathcal{M}})$ we deal with are endowed with a dimension $\dim(\mathcal{M})$, which we use to quantify running times of basic tasks, e.g., evaluating distances. We will mostly deal with metric spaces defined by a norm on $\mathbb{R}^d$, in which case $\dim(\mathcal{M}) = d$. We will assume that a point $x \in \mathcal{M}$ can be represented by $\mathrm{poly}(\dim(\mathcal{M}))$ bits, and that the distance $\mathrm{d}_{\mathcal{M}}(x, y)$ can be computed in time $\mathrm{poly}(\dim(\mathcal{M}))$, as well.

In this section we first introduce a formalization of the data-dependent LSH families we are going to use. We show how to use such LSH families to construct a data structure for the NNS problem, by generalizing the randomized decision tree data structure from [6]. Then we introduce the notion of weak average distortion embedding, and show that it can weak average distortion embeddings into ℓ_1^d or ℓ_2^d imply the existence of data-dependent LSH.

2.1 Data-dependent LSH Families Imply NNS

The following definition is standard.

▶ **Definition 8.** *Let $(\mathcal{M}, \mathrm{d}_{\mathcal{M}})$ be a metric space and fix a scale $r > 0$, approximation factor $c > 1$, and range Ω. Then a probability distribution $\mathcal{H}$ over maps from $\mathcal{M}$ to Ω is called (r, cr, p_1, p_2)-sensitive if*

$$\mathrm{d}_{\mathcal{M}}(x, y) \leq r \implies \mathop{\mathbb{P}}_{h \sim \mathcal{H}}[h(x) = h(y)] \geq p_1,$$
$$\mathrm{d}_{\mathcal{M}}(x, y) > cr \implies \mathop{\mathbb{P}}_{h \sim \mathcal{H}}[h(x) = h(y)] < p_2.$$

Our data structures are based on the following, in a sense, weaker definition which allows $\mathcal{H}$ to depend on the point set, and defines p_2 in terms of the how hash functions spread the points among the bins they are hashed to.

▶ **Definition 9.** *Let $(\mathcal{M}, \mathrm{d}_{\mathcal{M}})$ be a metric space and fix a scale $r > 0$, and range Ω. Let $P \subseteq \mathcal{M}$ be set of cardinality $|P| = n$. Then a probability distribution $\mathcal{H}(P)$ over maps from $\mathcal{M}$ to Ω is called (r, p_1, p_2)-empirically sensitive for P if*

$$\mathrm{d}_{\mathcal{M}}(x, y) \leq r \implies \mathop{\mathbb{P}}_{h \sim \mathcal{H}}[h(x) = h(y)] \geq p_1,$$
$$\forall \omega \in \Omega, \forall h \in \mathrm{supp}(\mathcal{H}): \quad |\{x \in P : h(x) = \omega\}| \leq p_2 n.$$

We call families of hash functions $\mathcal{H}(P)$ as in Definition 9 *data-dependent locality sensitive hash functions*, because p_2 constrains the data-dependent distribution of the points in P among the bins defined by the hash function.

▶ **Definition 10.** *Let $(\mathcal{M}, \mathrm{d}_{\mathcal{M}})$ be a metric space and fix a scale $t > 0$. A set P of n points in a metric space $\mathcal{M}$ is called (t, β)-dispersed if, for all $x \in \mathcal{M}$, $|P \cap B_{\mathcal{M}}(x, t)| \leq (1 - \beta)n$.*

The following is a straightforward observation.[4]

▶ **Lemma 11.** *Suppose for a set P of n points in a metric space $\mathcal{M}$, for every $x_0 \in P$ we have $|P \cap B_{\mathcal{M}}(x_0, 2t)| \leq (1 - \beta)n$. Then P is (t, β)-dispersed.*

▶ **Definition 12.** *For a metric space $(\mathcal{M}, \mathrm{d}_{\mathcal{M}})$, and a (multi-)set $P \subset \mathcal{M}$ of size n, we use the notation*

$$\Psi_{\mathcal{M}}(P, t) = \frac{|\{(x, y) \in P \times P : \mathrm{d}_{\mathcal{M}}(x, y) > t\}|}{n^2}.$$

The following lemma relates the notion of being (r, β)-dispersed and the function $\Psi_{\mathcal{M}}(P, t)$.

▶ **Lemma 13.** *Let $(\mathcal{M}, \mathrm{d}_{\mathcal{M}})$ be a metric space, and let P be a set of n points in $\mathcal{M}$. Then, if P is (t, β)-dispersed, then $\Psi_{\mathcal{M}}(P, t) \geq \beta$. Conversely, P is (t, β)-dispersed for $\beta = \frac{1}{2}\Psi_{\mathcal{M}}(P, 2t)$.*

The next lemma shows that data-dependently LSH families imply the existence of efficient NNS data structures.

▶ **Lemma 14.** *Let $(\mathcal{M}, \mathrm{d}_{\mathcal{M}})$ be a metric space and let $r > 0$, and $c > 1$. Suppose that for every $(cr, \frac{1}{2})$-dispersed m-point set $Q \subseteq \mathcal{M}$, there exists a $(r, p_1(Q), p_2(Q))$-empirically sensitive $\mathcal{H}(Q)$ such that $\frac{\log(1/p_1(Q))}{\log(1/p_2(Q))} \leq \rho$ where all $p_2(Q) \leq p_2$ for some $p_2 \in (0, 1)$. Define $b = \max(\frac{1}{2}, p_2)$. Further, suppose that any h in the support of $\mathcal{H}(Q)$ can be stored in space and evaluted in time polynomial in $\dim(\mathcal{M})$, that $h \sim \mathcal{H}(Q)$ can be sampled in time $T_s(m)$, for a non-decreasing function $T_s(m)$ and the range Ω of $\mathcal{H}(Q)$ has size at most $\exp(\dim(\mathcal{M}))$. Then there exists a data structure for the $(O(c), r)$-NNS problem over n-point sets in $\mathcal{M}$ with*

- *pre-processing time $O(\mathrm{poly}(n \log_{1/b}(n) \dim(\mathcal{M})) \cdot T_s(n))$;*
- *space $O(n^{1+\rho} \log_{1/b}(n) \cdot \mathrm{poly}(\dim(\mathcal{M})))$;*
- *query time $O(n^{\rho} \log_{1/b}(n) \cdot \mathrm{poly}(\dim(\mathcal{M})))$;*

The terminology used in its proof is largely adopted from [6]. The data structure is a randomized decision tree similar to the one in [6] with a couple of generalizations: we allow the hash function to split space into more than two parts, and, more importantly, we allow the parameters of the data-dependent LSH family to change from one node of the decision tree to the next.

[4] The proofs of the statements in this section can be found in the full version of this paper.

2.2 Data-dependent LSH Families from LSH Families

We show that the existence of an LSH family implies the existence of a data-dependent LSH family.

▶ **Lemma 15.** *Let $(\mathcal{M}, \mathrm{d}_{\mathcal{M}})$ be a metric space and let $r > 0$, and $c > 1$. Suppose there exists a (r, cr, p_1, p_2)-sensitive $\mathcal{H}$ where $p_2 \leq 1/2$. Let $P \subseteq \mathcal{M}$ be a (cr, β)-dispersed n-point set. Then, for*

$$p_2' = \sqrt{1 - \beta(1 - 2p_2)}$$

the event $\mathcal{E} = \{\max_{\omega \in \Omega} |\{x \in P : h(x) = \omega\}| \leq p_2' n\}$ occurs with probability at least $1/2$, and $\mathcal{H}$ conditioned on $\mathcal{E}$ is $(r, 2p_1 - 1, p_2')$-empirically sensitive for P.

We instantiate this lemma with the LSH for ℓ_1^d due to Indyk and Motwani [15] (see also [13]).

▶ **Lemma 16.** *For any $r > 0$, any $c > 1$, and any $\Delta \geq 1$, the space ℓ_1^d restricted to $[-\Delta, \Delta]^d$ has a (r, cr, p_1, p_2)-sensitive $\mathcal{H}$ with $p_1 = 1 - \frac{r}{2d\Delta}$ and $p_2 = 1 - \frac{cr}{2d\Delta}$. Moreover, $h \sim \mathcal{H}$ can be sampled and evaluated in constant time.*

Together with Lemma 15, Lemma 16 implies the following corollary.

▶ **Corollary 17.** *Let $r > 0$, $c > 6$, $\Delta \geq 1$, and let P be a (cr, β)-dispersed n-point set in ℓ_1^d restricted to $[-\Delta, \Delta]^d$. There exists a $(r, 1 - \frac{8}{c}, 1 - \frac{\beta}{4})$-empirically sensitive $\mathcal{H}_{\ell_1^d}(P)$ for P. Moreover, a function $h \sim \mathcal{H}(P)$ can be sampled in $O(n \operatorname{poly}(d\Delta/r))$ time, evaluated in $\operatorname{poly}(d\Delta/r)$ time, and stored using $\operatorname{poly}(d\Delta/r)$ bits.*

2.3 Weak Average Distortion Embeddings

Below is our definition of weak average distortion embedding. We will use such embeddings to construct data-dependent LSH families for spaces other than ℓ_1^d.

▶ **Definition 18.** *Let $(\mathcal{M}, \mathrm{d}_{\mathcal{M}})$ and $(\mathcal{N}, \mathrm{d}_{\mathcal{N}})$ be two metric spaces, and let $P \subseteq \mathcal{M}$ be an n-point set. A function $f : \mathcal{M} \to \mathcal{N}$ is an embedding with weak average distortion D with respect to P if we have*

$$\sup_{t \geq 0} t\Psi_{\mathcal{N}}(f(P), t) \geq \frac{\|f\|_{\mathrm{Lip}}}{D} \sup_{t \geq 0} t\Psi_{\mathcal{M}}(P, t).$$

The name "weak average distortion" originates from the fact that $\sup_{t \geq 0} t\Psi_{\mathcal{M}}(P, t)$ is the weak-L_1 norm of $\mathrm{d}_{\mathcal{M}}$ with respect to the uniform measure over $f(P) \times f(P)$. So, a weak average distortion embedding is required to not expand distances too much while also not decreasing the weak-L_1 norm of the pairwise distances. The analogous notion of q-average distortion, where we instead take the L_q norm of $\mathrm{d}_{\mathcal{M}}$ with respect to the same measure (see Definition 6), has been studied before. The definition of weak average distortion embedding appears to be new. It can be extended in the natural way to more general probability measures, but we will not pursue this here.

In this subsection, we show that a weak average distortion embeddings of $\mathcal{M}$ into ℓ_1^d imply, via Corollary 17, a data-dependent LSH family for $\mathcal{M}$.

▶ **Lemma 19.** *Let $r > 0$, $D \geq 1$, and $\Delta > 0$. Fix an approximation factor $c \geq 48D$. Suppose that P is a $(cr, \frac{1}{2})$-dispersed set of n points in a metric space $(\mathcal{M}, \mathrm{d}_{\mathcal{M}})$, and let Δ be the diameter of P. If $f : \mathcal{M} \to \ell_1^d$ (or $f : \mathcal{M} \to \ell_2^d$) is an embedding with weak average distortion D with respect to P, then there exists a (r, p_1, p_2)-empirically sensitive $\mathcal{H}(P)$ for P with $\frac{\log(1/p_1)}{\log(1/p_2)} \lesssim \frac{D}{c}$ and $p_2 \geq 1 - \frac{cr}{16D\Delta}$.*

Moreover, assume that $d \in \text{poly}(\dim(\mathcal{M}))$*, and that* f *can be computed from* P *in time* T*, and then stored in* $\text{poly}(\dim(\mathcal{M}))$ *bits, and evaluated in* $\text{poly}(\dim(\mathcal{M}))$ *time. Then a function* $h \sim \mathcal{H}(P)$ *can be sampled in time* $O(T + \text{poly}(n \dim(\mathcal{M})\Delta/r))$*, evaluated in* $\text{poly}(\dim(\mathcal{M})\Delta/r)$ *time, and stored using using* $\text{poly}(\dim(\mathcal{M})\Delta/r)$ *bits.*

3 From Average to Weak Average Distortion Embeddings

We show the following theorem connecting average and weak average distortion embeddings.

▶ **Theorem 20.** *Suppose that the metric space* $(\mathcal{M}, \mathrm{d}_{\mathcal{M}})$ *embeds with into a Banach space* $(X, \|\cdot\|)$ *with average distortion* D*. Then, for any* n*-point set* $P \subseteq \mathcal{M}$*, there exists an embedding with weak average distortion* $D' \lesssim D(1 + \log D)$*.*

Moreover, assume that for any point set Q *of* $m \leq n$ *points in* $\mathcal{M}$*, an embedding into* X *with average distortion* D *with respect to* Q *can be can be computed from* Q *in time* T*, and then stored in* $\text{poly}(\dim(\mathcal{M}))$ *bits, and evaluated in* $\text{poly}(\dim(\mathcal{M}))$ *time. Then, for any* n*-point set* $P \subseteq \mathcal{M}$*, the embedding into* X *with weak average distortion* D' *can be computed in time* $\text{poly}(T + n \dim(\mathcal{M}))$*, stored in* $\text{poly}(\dim(\mathcal{M}))$ *bits, and evaluated in time* $\text{poly}(\dim(\mathcal{M}))$*.*

The proof of Theorem 20 is inspired by arguments relating p-average and q-average distortion embeddings for different p and q in [26], and is discussed in complete detail in the full version of this paper.

4 Efficient Average Distortion Embeddings

4.1 Average Distortion Embeddings from Bi-Hölder Homeomorphisms

We first give a general construction of embeddings with bounded average distortion using homeomorphisms between spheres of normed spaces. The main theorems mentioned in the Introduction are then proved using this general construction and the classical and non-commutative Mazur maps as the homeomorphisms. Proofs and some of the additional theorems from this section can be found in the full version of this paper. For a Banach space $(X, \|\cdot\|)$, we will use the notation $S_X = \{x \in X : \|x\| = 1\}$.

We begin by showing that homeomorphisms between spheres can be radially extended to the entire normed spaces while retaining their continuity properties. The lemma below was also shown in [7] but with worse constants.

▶ **Lemma 21.** *Let* $(X, \|\cdot\|_X)$ *and* $(Y, \|\cdot\|_Y)$ *be Banach spaces, and let* $\alpha, \beta \in (0, 1]$*. Let* $f : S_X \to S_Y$ *be a function that, for any* $x, y \in S_X$ *satisfies*

$$\frac{1}{L}\|x - y\|_X^{1/\beta} \leq \|f(x) - f(y)\|_Y \leq K\|x - y\|_X^{\alpha}.$$

Then the function $\widetilde{f} : X \to Y$ *defined by* $\widetilde{f}(x) = \|x\|_X^{\alpha} f\left(\frac{x}{\|x\|_X}\right)$*, and* $\widetilde{f}(0) = 0$ *satisfies the following for any* $x, y \in X$*:*

$$\|\widetilde{f}(x) - \widetilde{f}(y)\|_Y \leq (1 + 2^{\alpha}K)\|x - y\|_X^{\alpha} \tag{1}$$

$$\|\widetilde{f}^{-1}(x) - \widetilde{f}^{-1}(y)\|_X \leq \left(\frac{1}{\alpha\beta} + 2^{\beta}L^{\beta}\right)\|x - y\|_Y^{\beta} \max\{\|x\|_Y, \|y\|_Y\}^{\frac{1}{\alpha} - \beta} \tag{2}$$

Moreover, $\|\widetilde{f}(x)\|_Y = \|x\|_X^{\alpha}$ *for all* $x \in X$*.*

Recall that a median of a set of n points P in a metric space $\mathcal{M}$ is any point $y \in \mathcal{M}$ that minimizes $\frac{1}{n}\sum_{x\in P} \mathrm{d}_{\mathcal{M}}(x,y)$. The next definition is an approximate version of the median.

▶ **Definition 22.** *We say that a point y in a metric space $\mathcal{M}$ is a (C,ε)-approximate median of a finite point set $P \subseteq \mathcal{M}$ if*

$$\frac{1}{n}\sum_{x\in P} \mathrm{d}_{\mathcal{M}}(x,y) \leq C \min_{z\in\mathcal{M}} \frac{1}{n}\sum_{x\in P} \mathrm{d}_{\mathcal{M}}(x,z) + \varepsilon.$$

The next lemma is our main tool for constructing explicit average distortion embeddings. Recall that, for $\alpha \in (0,1]$ the α-snowflake of a metric space $(\mathcal{M}, \mathrm{d}_{\mathcal{M}})$ is the metric space $\mathcal{M}^\alpha$ on the same ground set, with distance function $\mathrm{d}_{\mathcal{M}}(x,y)^\alpha$.

▶ **Lemma 23.** *Let $(X, \|\cdot\|_X)$ and $(Y, \|\cdot\|_Y)$ be Banach spaces, and let $\alpha \in (0,1]$. Let $f : S_X \to S_Y$ be a function that, for any $x, y \in S_X$ satisfies*

$$\|f(x) - f(y)\|_Y \leq K\|x-y\|_X^\alpha,$$

and let $\widetilde{f}(x) = \|x\|_X^\alpha f\left(\frac{x}{\|x\|_X}\right)$, $\widetilde{f}(0) = 0$. Let $P \subseteq X$ be an n point set, let $t \in X$, and define $g : X^\alpha \to Y$ by $g(x) = \widetilde{f}(x-t)$. Suppose that one of the following conditions is satisfied for some $C \geq 1$ and $\varepsilon \leq \frac{1}{4n^2}\sum_{x\in P}\sum_{y\in P}\|x-y\|_X^\alpha$

1. *0 is a (C,ε)-approximate median for $g(P)$;*
2. *$\left\|\frac{1}{n}\sum_{x\in P} g(x)\right\|_Y \leq \varepsilon$.*

Then g is an embedding of the α-snowflake X^α into Y with average distortion at most D with respect to P, where $D \lesssim C(1+K)$ if the first condition is satisfied, and $D \lesssim 1+K$ if the second condition is satisfied.

Its proof is similar in spirit to bounds on non-linear Rayleigh quotients proved in [7], and is discussed in the full version. In order to use Lemma 23, we need to find some $t \in X$ that satisfies one of the two assumptions in the lemma. A general method for establishing the existence of such a t was proposed in [6, 7, 26], and relies on the following lemma. For a proof, see Lemma 45 from [26].

▶ **Lemma 24.** *For any finite dimensional Banach space $(X, \|\cdot\|)$, and any continuous function $h : X \to X$ such that*

$$\lim_{M\to\infty} \inf_{t:\|t\|\geq M} (\|t\| - \|h(t)-t\|) = \infty,$$

we have that h is surjective.

Using Lemma 24, we can show that there exists a t so that 0 is the mean of $g(P)$. The argument is essentially identical to arguments in [6, 7], but, since the result was not stated in the general form given below, we include a proof of the following lemma in the full version.

▶ **Lemma 25.** *Under the assumptions and notation of Lemma 21, there exists some $t \in X$ such that $\frac{1}{n}\sum_{x\in P} g(x) = 0$. Moreover, any such t must satisfy $\|t\|_X \leq \left(\frac{M}{n}\sum_{x\in P}\|x\|_X^\alpha\right)^{\frac{1}{\alpha}}$, for a constant M that only depends on K, L, α, β.*

While Lemma 25 is very general, it does not readily give rise to an efficient algorithm to find t. The proof of Lemma 24 in [6, 26] is existential, and relies on a topological degree argument of the type used to prove Brouwer's fixed point theorem. Identifying general cases in which we can give an algorithmic proof of Lemma 25 is an interesting open problem. In the full version of this paper, we instantiate Lemma 23 with the Mazur map (for embedding ℓ_p^d) and the non-commutative Mazur map (for embedding Schatten-p), and give alternative algorithmic methods for finding a good center t in these special cases. This leads us to the proofs of Theorems 3, 4, and 5 stated in the Introduction.

5 Conclusion and Open Problems

We have constructed data structures for the (c, r)-NNS problem with efficient pre-processing, nearly linear space, and sub-linear query time with approximation $c \lesssim p$ in the case of ℓ_p spaces for all $p \geq 1$, and with $c \lesssim 1$ for Schatten-p spaces for $1 \leq p \leq 2$. Furthermore, we have laid out a general framework for producing such efficient data structures for general metrics: as long as there are (computationally efficient) average distortion embeddings of such metrics into ℓ_1^d or ℓ_2^d, we can produce efficient NNS data structures. This framework is an analogue of the cutting modulus framework from [6], but allows efficient pre-processing.

This connection between NNS data structures and low average distortion embeddings naturally warrants further research into constructing such computationally efficient embeddings of arbitrary metric spaces into ℓ_1^d or ℓ_2^d. A natural first step is to do this for Schatten-p spaces where $p > 2$. As noted earlier, the bottleneck in our construction is the design of an efficient algorithm for computing a center T satisfying the conditions of Lemma 23 for embedding Schatten-p into Schatten-2.

▶ **Problem 1.** *Given a dataset P of n $d \times d$ symmetric matrices in Schatten-p, find a matrix T in* $\mathrm{poly}(n, d, 1/\varepsilon)$ *time such that either* 0 *is a (C, ε)-median of $\widetilde{M}_{p,2}(T - P)$, or* $\left\| \frac{1}{n} \sum_{X \in P} \widetilde{M}_{p,2}(T - X) \right\|_{C_2} \leq \varepsilon$.

In our quest to construct efficient NNS data structures for arbitrary finite-dimensional norms, a slightly more ambitious goal is to make the main result of [26] algorithmic, as follows.

▶ **Problem 2.** *Given an n-point dataset P in a d-dimensional normed space $(\mathcal{M}, \|\cdot\|)$, construct an embedding $f : \mathcal{M}^{\frac{1}{2}} \to \ell_2^d$ with 2-average distortion $\lesssim \sqrt{\log d}$ with respect to P such that f can be computed in time* $\mathrm{poly}(nd)$*, stored in poly(d) bits, and evaluated in time* $\mathrm{poly}(d)$.

A solution to this problem will imply NNS data structures for any d-dimensional norm with polynomial time pre-processing, nearly linear space, sub-linear query time, and approximation poly-logarithmic in the dimension, solving also an open problem in [6]. Note that such data structures are not known even with exponential pre-processing, but it was shown in [6] that they do exist in the cell-probe model.

Finally, on a somewhat different note, it would also be very interesting to further optimize the approximation factor c of our NNS data structures, even in the special case of ℓ_p spaces.

▶ **Problem 3.** *Establish Theorem 3 with $c \lesssim \frac{\log p}{\varepsilon}$.*

A solution to this problem would interpolate between data structures for ℓ_1^d and ℓ_2^d, where constant approximation is possible, and Indyk's data structure for ℓ_∞^d which guarantees an $O(\log \log d)$ approximation [14], and is optimal in several natural models [1].

References

1 Alexandr Andoni, Dorian Croitoru, and Mihai Patrascu. Hardness of Nearest Neighbor under L-infinity. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS '2008)*, pages 424–433, 2008.

2 Alexandr Andoni and Piotr Indyk. Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS '2006)*, pages 459–468, 2006.

3 Alexandr Andoni, Piotr Indyk, Huy L. Nguyen, and Ilya Razenshteyn. Beyond Locality-Sensitive Hashing. In *Proceedings of the 25th ACM-SIAM Symposium on Discrete Algorithms (SODA '2014)*, pages 1018–1028, 2014. Available as `arXiv:1306.1547`.

4 Alexandr Andoni, Piotr Indyk, and Ilya Razenshteyn. Approximate nearest neighbor search in high dimensions. In *Proceedings of ICM 2018 (to appear)*, 2018.

5 Alexandr Andoni, Thijs Laarhoven, Ilya Razenshteyn, and Erik Waingarten. Optimal Hashing-based Time–Space Trade-offs for Approximate Near Neighbors. In *Proceedings of the 28th ACM-SIAM Symposium on Discrete Algorithms (SODA '2017)*, pages 47–66, 2017. Available as `arXiv:1608.03580`.

6 Alexandr Andoni, Assaf Naor, Aleksandar Nikolov, Ilya P. Razenshteyn, and Erik Waingarten. Data-dependent hashing via nonlinear spectral gaps. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 787–800. ACM, 2018. `doi:10.1145/3188745.3188846`.

7 Alexandr Andoni, Assaf Naor, Aleksandar Nikolov, Ilya P. Razenshteyn, and Erik Waingarten. Hölder homeomorphisms and approximate nearest neighbors. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 159–169. IEEE Computer Society, 2018. `doi:10.1109/FOCS.2018.00024`.

8 Alexandr Andoni and Ilya Razenshteyn. Optimal Data-Dependent Hashing for Approximate Near Neighbors. In *Proceedings of the 47th ACM Symposium on the Theory of Computing (STOC '2015)*, pages 793–801, 2015. Available as `arXiv:1501.01062`.

9 Yair Bartal and Lee-Ad Gottlieb. Approximate nearest neighbor search for ℓ_p-spaces ($2<p<\infty$). *Theor. Comput. Sci.*, 757:27–35, 2019. `doi:10.1016/j.tcs.2018.07.011`.

10 S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509–522, April 2002. `doi:10.1109/34.993558`.

11 Yoav Benyamini and Joram Lindenstrauss. *Geometric Nonlinear Functional Analysis. Vol. 1*, volume 48 of *American Mathematical Society Colloquium Publications*. American Mathematical Society, Providence, RI, 2000.

12 Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. Locality-Sensitive Hashing Scheme Based on p-Stable Distributions. In *Proceedings of the 20th ACM Symposium on Computational Geometry (SoCG '2004)*, pages 253–262, 2004.

13 Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity Search in High Dimensions via Hashing. In *Proceedings of the 25th International Conference on Very Large Data Bases (VLDB '1999)*, pages 518–529, 1999.

14 Piotr Indyk. On Approximate Nearest Neighbors under ℓ_∞ Norm. *Journal of Computer and System Sciences*, 63(4):627–638, 2001.

15 Piotr Indyk and Rajeev Motwani. Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. In *Proceedings of the 30th ACM Symposium on the Theory of Computing (STOC '1998)*, pages 604–613, 1998.

16 Yi Li, Huy L. Nguyên, and David P. Woodruff. On Sketching Matrix Norms and the Top Singular Vector. In *Proceedings of the 25th ACM-SIAM Symposium on Discrete Algorithms (SODA '2014)*, pages 1562–1581, 2014.

17 Yi Li and David P. Woodruff. On Approximating Functions of the Singular Values in a Stream. In *Proceedings of the 48th ACM Symposium on the Theory of Computing (STOC '2016)*, pages 726–739, 2016.

18 Yi Li and David P. Woodruff. Embeddings of Schatten Norms with Applications to Data Streams. In *Proceedings of the 44th International Colloquium on Automata, Languages and Programming (ICALP '2017)*, pages 60:1–60:14, 2017.

19 Richard J. Lipton and Robert Endre Tarjan. Applications of a planar separator theorem. *SIAM J. Comput.*, 9(3):615–627, 1980. `doi:10.1137/0209046`.

20 Jiří Matoušek. On Embedding Expanders into ℓ_p Spaces. *Israel Journal of Mathematics*, 102:189–197, 1997.

21 Stefan Meiser. Point location in arrangements of hyperplanes. *Inf. Comput.*, 106(2):286–303, 1993. doi:10.1006/inco.1993.1057.

22 Bartlett W. Mel. Seemore: Combining color, shape, and texture histogramming in a neurally inspired approach to visual object recognition. *Neural Computation*, 9(4):777–804, 1997. doi:10.1162/neco.1997.9.4.777.

23 Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.

24 Renée J. Miller, Fatemeh Nargesian, Erkang Zhu, Christina Christodoulakis, Ken Q. Pu, and Periklis Andritsos. Making open data transparent: Data discovery on open data. *IEEE Data Eng. Bull.*, 41(2):59–70, 2018. URL: http://sites.computer.org/debull/A18june/p59.pdf.

25 Assaf Naor. Comparison of Metric Spectral Gaps. *Analysis and Geometry in Metric Spaces*, 2:1–52, 2014.

26 Assaf Naor. An average John theorem. *arXiv preprint*, 2019. arXiv:1905.01280.

27 Assaf Naor, Gilles Pisier, and Gideon Schechtman. Impossibility of Dimension Reduction in the Nuclear Norm. In *Proceedings of the 29th ACM-SIAM Symposium on Discrete Algorithms (SODA '2018)*, 2018.

28 Assaf Naor and Yuval Rabani. On Approximate Nearest Neighbor Search in ℓ_p, $p > 2$. Manuscript, available on request, 2006.

29 Éric Ricard. Hölder Estimates for the Noncommutative Mazur Map. *Archiv der Mathematik*, 104(1):37–45, 2015.

30 Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theor. Comput. Sci.*, 348(2-3):357–365, 2005. doi:10.1016/j.tcs.2005.09.023.

Convergence of Gibbs Sampling: Coordinate Hit-And-Run Mixes Fast

Aditi Laddha
Georgia Institute of Technology, Atlanta, GA, USA

Santosh S. Vempala
Georgia Institute of Technology, Atlanta, GA, USA

Abstract

The Gibbs Sampler is a general method for sampling high-dimensional distributions, dating back to 1971. In each step of the Gibbs Sampler, we pick a random coordinate and re-sample that coordinate from the distribution induced by fixing all the other coordinates. While it has become widely used over the past half-century, guarantees of efficient convergence have been elusive. We show that for a convex body K in $\mathbb{R}^n$ with diameter D, the mixing time of the Coordinate Hit-and-Run (CHAR) algorithm on K is polynomial in n and D. We also give a lower bound on the mixing rate of CHAR, showing that it is strictly worse than hit-and-run and the ball walk in the worst case.

2012 ACM Subject Classification Mathematics of computing → Markov processes

Keywords and phrases Gibbs Sampler, Coordinate Hit and run, Mixing time of Markov Chain

Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.51

Funding *Aditi Laddha*: Supported in part by NSF awards 1839323 and 2007443.
Santosh S. Vempala: Supported in part by NSF awards 1839323, 1909756 and 2007443.

1 Introduction

Sampling a high-dimensional distribution is a fundamental problem and a basic ingredient of algorithms for optimization, integration, statistical inference, and other applications. Progress on sampling algorithms has led to many useful tools, both theoretical and practical. In the most general setting, given access to a function $f : \mathbb{R}^n \to \mathbb{R}_+$, the goal is to generate a point x whose density is proportional to $f(x)$. Two special cases of particular interest are when f is uniform over a convex body and when f is a Gaussian restricted to a convex set.

The generic approach to sampling is by a Markov chain whose state space is the convex body. The chain is designed so that it is ergodic, time-reversible, and has the desired density as its stationary distribution. The key question is to bound the rate of convergence of the Markov chain. The Ball walk [14, 12, 18] and Hit-and-Run [2, 22, 17] are two Markov chains that work in full generality, and have been shown to mix rapidly (i.e, the convergence rate is polynomial) for arbitrary log-concave densities. Over three decades of improvements, the complexity of this problem has been reduced to a small polynomial in the dimension for the total number of function evaluations with a factor of n^2 per function call for the total number of arithmetic operations. For a log-concave density with support of diameter D, the mixing time is $O^*(n^2D^2)$, taking the same number of function evaluations, with arithmetic complexity of $O^*(n^4D^2)$[12, 15, 17].

A simple and widely-used algorithm that pre-dates these developments considerably is the Gibbs Sampler, proposed by Turchin in 1971 [23]. It is inspired by statistical physics and is commonly used for sampling distributions [5, 6] and Bayesian inference [8, 9, 10]. To sample a multivariate density, at each step, the sampler picks a coordinate (either at random or in order, cycling through the coordinates), fixes all other coordinates, and re-samples this coordinate from the induced distribution. This is very similar to Hit-and-Run, except that instead of picking a direction uniformly at random from the unit sphere, it is picked only

from one of the n basis vectors (see [1] for a historical account and more background). It was reported to be significantly faster than Hit-and-Run in state-of-the-art software for volume computation and integration [3, 7, 4]. Gibbs sampling, also called Coordinate Hit-and-Run, has a computational benefit: updating the current point takes $O(n)$ time rather than $O(n^2)$ even for polyhedra since the update is along only one coordinate direction. Thus the overhead per step is reduced from $O(n^2)$ as in all previous algorithms to $O(n)$. However, despite half a century of intense study, the convergence rate of Gibbs sampling has remained an open problem.

In this paper, we show that the Gibbs sampler mixes rapidly for any convex body K. Before stating our main theorem formally, we define the Gibbs sampler.

Coordinate Hit-and-Run

Algorithm 1 describes the Coordinate Hit-and-Run walk for sampling uniformly from a convex body $K \in \mathbb{R}^n$. Let $\{\mathbf{e}_i : 1 \leq i \leq n\}$ be the standard basis for $\mathbb{R}^n$. The starting point is in the interior of K and is given as an input to the algorithm.

Algorithm 1 Coordinate Hit-and-Run (CHAR).

Input: a point $x^{(0)} \in K$, integer T.
for $i = 1, 2, \cdots, T$ **do**
  Pick a uniformly random axis direction e_j
  Set x^i to be a random point along the line $\ell = \left\{x^{(i-1)} + te_j \,:\, t \in \mathbb{R}\right\}$ chosen uniformly from $\ell \cap K$.
end
Output: x^T.

The stationary distribution of the Coordinate hit-and-run walk is the uniform distribution π_K over K. To sample from a general log-concave density $f : \mathbb{R}^n \to \mathbb{R}_+$ the only change is in Step 2, where the next point y is chosen according to $f(y)$ restricted to ℓ. In both cases, the process is symmetric and ergodic and so the stationary distribution of the Markov chain is the desired distribution.

We can now state our main theorem (see Sec. 1.2 for the definition of a warm start).

▶ **Theorem 1.** *Let K be a convex body in $\mathbb{R}^n$ containing a unit ball. Let R^2 be the expected squared distance of a uniform random point in K from the centroid of K. Then the mixing time of Coordinate Hit-and-Run from a warm start in K is $\widetilde{O}\left(n^9 R^2\right)$.*

By applying an affine transformation, R can be made $O(\sqrt{n})$. We note that from a warm start both the Ball Walk and Hit-and-Run have a mixing time of $\widetilde{O}(n^2R^2)$ [12, 17]. While our bound is likely not the best polynomial bound for CHAR, in Section 4, we show that it is necessarily higher than the bound for hit-and-run.

Concurrently and independently, Narayanan and Srivastava [20] also proved a polynomial bound on the mixing rate of Coordinate Hit-and-Run, with a different proof. They showed that CHAR mixes in $\tilde{O}(n^7 R_1^4)$ steps where R_1 is the smallest number s.t., $B_\infty \subseteq K \subseteq R_1 B_\infty$, i.e., the cube sandwiching ratio (R_1 can be larger than R in our theorem by a factor of $\sqrt{n}$). This quantity R_1 can be bounded by $O(n)$ after an affine transformation.

A key ingredient of our proof is a new "ℓ_0"-isoperimetric inequality. We will need the following definition.

▶ **Definition 2** (Axis-disjoint). *Two measurable sets S_1, S_2 are called* axis-disjoint *if* $\forall x \in S_1, \forall y \in S_2, |\{i \in [n] : x_i = y_i\}| \leq n - 2$.

In other words, no point from S_1 is on the same axis-parallel line as any point in S_2 (see Fig. 1).

The main component of the proof of Theorem 1 is the following isoperimetric inequality for axis-disjoint subsets of a convex body.

▶ **Theorem 3.** *Let K be a convex body in $\mathbb{R}^n$ containing a unit ball with $R^2 = \mathbb{E}_K(\|x - z_K\|^2)$ where z_K is the centroid of K. Let $S_1, S_2 \subseteq K$ be two measurable subsets of K such that S_1, S_2 are axis-disjoint. Then for any $\epsilon \geq 0$, the set $S_3 = K \backslash \{S_1 \cup S_2\}$ satisfies*

$$\mathrm{vol}(S_3) \geq \frac{\epsilon}{48 \cdot 10^3 \cdot n^{3.5} R \log n} \left(\min\{\mathrm{vol}(S_1), \mathrm{vol}(S_2)\} - \epsilon \mathrm{vol}(K)\right).$$

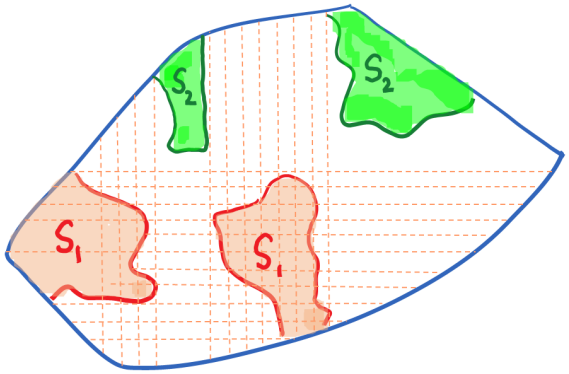


Figure 1 Axis-disjoint subsets S_1 and S_2.

1.1 Approach

At a high level, we follow the proof of rapid mixing based on the conductance of Markov chains [21] in the continuous setting [16]. We give a simple, new one-step coupling lemma which reduces the problem of lower bounding the conductance of the underlying Markov chain to an isoperimetric inequality about axis-disjoint sets in high dimension. Roughly speaking, the inequality says the following: If two subsets of a convex body are axis-disjoint, then the remaining mass of the body is proportional to the smaller of the two subsets. This inequality is our main technical contribution. In comparison, the inequality for Euclidean distance says that for any two subsets of a convex body, the remaining mass is proportional to their (minimum) Euclidean distance times the smaller of the two subset volumes.

Standard approaches to proving such inequalities, notably localization [11, 13], which reduce the desired high-dimensional inequality to a one-dimensional inequality, do not seem to be directly applicable to proving this "ℓ_0-type" inequality. So we develop a first-principles approach where we first prove the inequality for cubes, taking advantage of their product structure, and then for general bodies using a tiling of space with cubes. In the course of the latter part, we will use several known properties of convex bodies, including Euclidean isoperimetry.

1.2 Preliminaries

- **Markov chain**: Let $\mathcal{M}$ be a Markov chain with state space K and stationary distribution Q. For any measurable subset $S \subseteq K$ and $x \in K$, let $P_x(S)$ be the probability that one step of $\mathcal{M}$ from x goes to a point in S. Q being stationary implies that for any measurable subset $S \subseteq K$

  $$\int_K P_x(S) dQ(x) = Q(S).$$

  The Markov chain is **time-reversible** if for any two subsets $A, B \subseteq K$

  $$\int_A P_x(B) dQ(x) = \int_B P_x(A) dQ(u).$$

  For a measurable subset S of the state space of a Markov chain with stationary distribution Q, the ergodic flow of S, denoted by $p(S)$ is defined as

  $$p(S) = \int_S P_x(K \setminus S)\, dQ(x).$$

- **Conductance**: The conductance of a subset $S \subseteq K$, denoted by $\phi(S)$, is defined as

  $$\phi(S) = \frac{\int_S P_x(K \setminus S)\, dQ(x)}{\min\{Q(S), Q(K \setminus S)\}},$$

  and the conductance of $\mathcal{M}$ is defined as

  $$\phi = \inf_{0 < Q(S) \leq 1/2} \phi(S).$$

  For any $s \in [0, 1/2]$ the s-conductance of the Markov chain is:

  $$\phi_s = \inf_{S : s < Q(S) \leq \frac{1}{2}} \frac{p(S)}{Q(S) - s}.$$

- **Warm Start**: Given distributions P and Q on the same state space $\mathcal{A}$, P is said to be M-warm with respect to Q if

  $$M = \sup_{A \subseteq \mathcal{A}} \frac{P(A)}{Q(A)}.$$

  If the initial distribution Q_0 is $O(1)$-warm with respect to the stationary distribution Q for some Markov chain, we say that Q_0 is a warm start for Q.
- **Lazy chain**: A lazy version of a Markov chain with transition probability P is one where we use the transition probability $P_x(\{y\}) = P_x(\{y\})/2 + \mathbf{1}(x = y)/2$, so that with probability $1/2$, the chain feels lazy and stays in the same state.
- For a body $K \subseteq \mathbb{R}^n$, let π_K denote the uniform distribution on K and $\mathbb{E}_K(X)$ denote the expected value of X with respect to π_K.

The following theorem shows that the s-conductance of a Markov chain bounds its rate of convergence from a warm start.

▶ **Theorem 4** ([16]). *Suppose that a lazy, time-reversible Markov chain with stationary distribution Q has s-conductance at least ϕ_s. Then with initial distribution Q_0, and*

$$H_s = \sup \{|Q(A) - Q_0(A)| : A \subset K, Q(A) \leq s\},$$

the distribution Q_t after t steps satisfies

$$d_{TV}(Q_t, Q) \leq H_s + \frac{H_s}{s}\left(1 - \frac{\phi_s^2}{2}\right)^t.$$

2 The isoperimetric inequality

Before proving Theorem 3, we need a few definitions.

▶ **Definition 5** (Axis-aligned Line). *A line ℓ in $\mathbb{R}^n$ is called* axis-aligned *if $\ell = \{x \in \mathbb{R}^n : x = c + te_i, t \in \mathbb{R}\}$ for an $i \in \{1, \ldots, n\}$ and a point $c \in \mathbb{R}^n$.*

▶ **Definition 6** (Axis-aligned Cube). *An axis-aligned cube C is defined as*

$$C = \{x \in \mathbb{R}^n : \|x - y\|_\infty \leq c\},$$

where $y \in \mathbb{R}^n$ is a fixed point and c is a positive constant.

▶ **Definition 7** (Axis-parallel Extension). *For a body K in $\mathbb{R}^n$ and a measurable subset $S \subseteq K$, the axis-parallel extension of S in K, denoted by $ext_K(S)$ is defined as*

$$\mathrm{ext}_K(S) = \{x \in K \backslash S : \exists y \in S \text{ such that } |\{i \in [n] : x_i = y_i\}| = n - 1\}.$$

In other words, $ext_K(S)$ is the set of points in $K \backslash S$ obtained by changing exactly 1 coordinate from a point in S.

We will bound the conductance of CHAR in an axis aligned cube by its mixing time.

▷ Claim 8. The mixing time of the Coordinate Hit-and-Run chain in an axis-aligned cube $C \in \mathbb{R}^n$ is $O(n \log n)$.

Proof. WLOG, assume that C is a cube with side length 1. In a step of CHAR, if axis e_i is selected, then the i-th coordinate of the current point is re-sampled uniformly at random from $[0, 1]$. So starting from any point in C, after every axis direction has been picked at least once, the distribution induced by CHAR will be uniform on C. Using the coupon collector bound, the number of steps of CHAR needed to ensure that every coordinate has been re-sampled at least once is at most $4n \log n = O(n \log n)$ in expectation. ◁

▶ **Lemma 9** (Cube isoperimetry). *For an axis-aligned cube $C \in \mathbb{R}^n$, and any two axis-disjoint subsets $S_1, S_2 \subseteq C$, with $S_3 = C \setminus \{S_1 \cup S_2\}$, the following holds:*

$$\mathrm{vol}(S_3) \geq \frac{1}{4n \log n} \cdot \min\{\mathrm{vol}(S_1), \mathrm{vol}(S_2)\}.$$

▶ Remark 10. We believe that the bound above is not optimal, and even an absolute constant factor might be possible. In the appendix, we give a different proof achieving a weaker bound.

Proof. Let $\mathrm{vol}(S_1) \leq \mathrm{vol}(S_2)$. For a Markov chain with mixing time t_{mix}, the conductance is at least $1/t_{mix}$ [19]. From Claim 8, we get

$$\phi(S_1) = \frac{\int_{S_1} P_x(C \setminus S_1)\, dx}{\mathrm{vol}(S_1)} \geq \frac{1}{4n \log n}$$

$$\int_{S_1} P_x(C \setminus S_1)\, dx \geq \frac{\mathrm{vol}(S_1)}{4n \log n} \tag{1}$$

Since S_1 and S_2 are axis-disjoint, the probability of moving from a point in S_1 to a point in S_2 in one step is 0 and hence

$$\int_{S_1} P_x(C \setminus S_1)\, dx \leq \int_{S_1} P_x(S_3)\, dx = \int_{S_3} P_x(S_1)\, dx \leq \mathrm{vol}(S_3). \tag{2}$$

Combining equation (1) and equation (2), we get

$$\mathrm{vol}(S_3) \geq \frac{1}{4n \log n} \cdot \mathrm{vol}(S_1).$$ ◀

The next lemma is an isoperimetric inequality from [11].

▶ **Lemma 11** (Euclidean isoperimetry). *[11] Let $K \subset \mathbb{R}^n$ be a convex body containing a unit ball and $R^2 = \mathbb{E}_K(\|x - z_k\|^2)$ where z_K is the centroid of K. For a subset $S \subseteq K$, let $\partial_K(S)$ denote the boundary of S, relative to K. Then for any $S \subseteq K$ of volume at most $\mathrm{vol}(K)/2$, we have*

$$\mathrm{vol}_{n-1}(\partial_K S) \geq \frac{\ln 2}{R}\mathrm{vol}(S).$$

We can now prove the new isoperimetric inequality, restated below for convenience.

▶ **Theorem 3.** *Let K be a convex body in $\mathbb{R}^n$ containing a unit ball with $R^2 = \mathbb{E}_K(\|x - z_K\|^2)$ where z_K is the centroid of K. Let $S_1, S_2 \subseteq K$ be two measurable subsets of K such that S_1, S_2 are axis-disjoint. Then for any $\epsilon \geq 0$, the set $S_3 = K \backslash \{S_1 \cup S_2\}$ satisfies*

$$\mathrm{vol}(S_3) \geq \frac{\epsilon}{48 \cdot 10^3 \cdot n^{3.5} R \log n} \left(\min\{\mathrm{vol}(S_1), \mathrm{vol}(S_2)\} - \epsilon \mathrm{vol}(K)\right).$$

Proof of Theorem 3. Let $K' = (1-\alpha)K$ for $\alpha = \frac{\epsilon}{20n}$, and $S_i' = S_i \cap K'$ for $i \in \{1, 2\}$. Assume $\mathrm{vol}(S_1') \leq \mathrm{vol}(S_2')$. For any subset $X \subseteq K$, we have

$$\mathrm{vol}(X \cap K') \geq \mathrm{vol}(X) - (1 - (1-\alpha)^n)\mathrm{vol}(K) \geq \mathrm{vol}(X) - \frac{\epsilon}{2} \cdot \mathrm{vol}(K).$$

Next consider a standard lattice of width δ, with each lattice point inducing a cube of side length δ. We choose $\delta = \frac{\alpha}{4\sqrt{n}}$ to ensure that the cubes which intersect K' are fully contained in K. Let $\mathcal{C}$ be the set of hypercubes in this lattice that intersect S_1. We partition the set of cubes in $\mathcal{C}$ as

- $\mathcal{C}_1$: the set of cubes in $\mathcal{C}$ where S_1 takes up less than $\frac{2}{3}$ of the volume of the cube, and
- $\mathcal{C}_2$: the set of cubes in $\mathcal{C}$ where S_1 takes up at least $\frac{2}{3}$ of each cube.

If $\mathrm{vol}(\mathcal{C}_1 \cap S_1) \geq \mathrm{vol}(S_1)/2$, i.e., at least $\frac{1}{2}$ of $\mathrm{vol}(S_1)$ resides in cubes in $\mathcal{C}_1$, then we can apply Lemma 9 to every cube in $\mathcal{C}_1$ individually to get a bound on $\mathrm{vol}(S_3)$. However, the cubes in $\mathcal{C}_1$ might not be fully contained in K and so before using the cube isoperimetry, we need to move to K' and consider the subset of cubes of $\mathcal{C}_1$ that intersect K'. Let $\mathcal{C}_1' = \{c \in \mathcal{C}_1 : c \cap K' \neq \emptyset\}$. By our choice of α, we have $\mathcal{C}_1' \subseteq K$ and

$$\mathrm{vol}(\mathcal{C}_1' \cap S_1) \geq \mathrm{vol}(\mathcal{C}_1 \cap S_1 \cap K') \geq \mathrm{vol}(\mathcal{C}_1 \cap S_1) - \frac{\epsilon}{2} \cdot \mathrm{vol}(K).$$

Now consider a cube c in $\mathcal{C}_1'$. Let $x = \frac{\mathrm{vol}(c \cap S_1)}{\mathrm{vol}(c)}$ and $y = \frac{\mathrm{vol}(c \cap S_2)}{\mathrm{vol}(c)}$ where $0 < x \leq 2/3$ and $x + y \leq 1$. Then, applying Lemma 9for any feasible values of x and y, we have

$$\frac{\mathrm{vol}(S_3 \cap c)}{\mathrm{vol}(c)} \geq \max\left\{1 - x - y, \frac{1}{4n \log n} \min\{x, y\}\right\} \geq \frac{1}{4n \log n} \cdot \frac{x}{4}$$

Applying this argument to each cube in $\mathcal{C}_1'$, we get

$$\begin{aligned}
\mathrm{vol}(S_3) &\geq \frac{1}{16n \log n} \cdot \sum_{c \in \mathcal{C}_1'} \mathrm{vol}(c \cap S_1) = \frac{1}{16n \log n} \cdot \mathrm{vol}(\mathcal{C}_1' \cap S_1) \\
&\geq \frac{1}{16n \log n} \left(\mathrm{vol}(\mathcal{C}_1 \cap S_1) - \frac{\epsilon}{2} \cdot \mathrm{vol}(K)\right) \\
&\geq \frac{1}{16n \log n} \left(\frac{1}{2} \cdot \mathrm{vol}(S_1) - \frac{\epsilon}{2} \cdot \mathrm{vol}(K)\right) \\
&\geq \frac{1}{32n \log n} \left(\mathrm{vol}(S_1) - \epsilon \cdot \mathrm{vol}(K)\right).
\end{aligned}$$

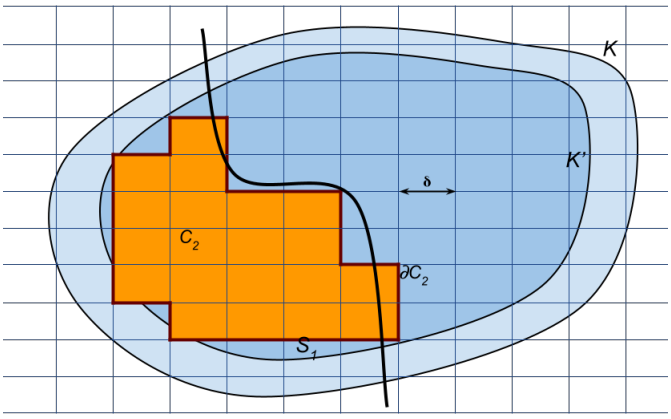


Figure 2 Illustration of the isoperimetry proof.

So assume that $\text{vol}(\mathcal{C}_1 \cap S_1) \leq \frac{1}{2} \cdot \text{vol}(S_1)$ and consequently $\text{vol}(\mathcal{C}_2 \cap S_1) \geq \frac{1}{2} \cdot \text{vol}(S_1)$. Let $\partial\mathcal{C}_2$ be the internal boundary of $\mathcal{C}_2$ in K. Because $\mathcal{C}_2$ is comprised of hypercubes, $\partial\mathcal{C}_2$ consists of facets of axis-aligned n-dimensional cubes. Consider a facet f on this boundary with normal axis e_f, let the cube adjacent to this facet in $\mathcal{C}_2$ be f_2 and the cube adjacent to the facet not in $\mathcal{C}_2$ be f_1. So, $\text{vol}_{n-1}(f) \cdot \delta = \text{vol}(f_1)$ and f_1 cannot be in $\mathcal{C}_2$ (it can belong to $\mathcal{C}_1$ but we account for that in the next step). Since at least $2/3$ of the volume f_2 is in S_1, the support of marginal of S_1 along any axis direction will be at least $2/3$ of the support of marginal of f_2 along that axis. Therefore, at least $2/3$ of the mass of f and by extension f_1 is reachable from a point in S_1 along e_f and therefore cannot be in S_2. Now if $f_1 \in \mathcal{C}_1$, there are 2 possibilities:

1. $\text{vol}(f_1 \cap S_1) \leq \text{vol}(f_1)/3$, we can simply subtract this volume from the volume of S_3 gained from f_1 and get
$$\text{vol}(f_1 \cap S_3) \geq \frac{2}{3} \cdot \delta \text{vol}_{n-1}(f) - \frac{1}{3} \cdot \text{vol}(f_1) = \frac{2}{3} \cdot \text{vol}(f_2) - \frac{1}{3} \cdot \text{vol}(f_1) = \frac{1}{3} \cdot \text{vol}(f_1).$$
2. Otherwise, $\frac{1}{3} \cdot \text{vol}(f_1) \leq \text{vol}(f_1 \cap S_1) \leq \frac{2}{3} \cdot \text{vol}(f_1)$ and using Lemma 9, we get
$$\text{vol}(f_1 \cap S_3) \geq \frac{1}{4n \log n} \cdot \frac{\text{vol}(f_1 \cap S_1)}{4} = \frac{1}{48n \log n} \cdot \text{vol}(f_1).$$

So, for every facet $f \in \partial_K(\mathcal{C}_2)$, we get

$$\min\left\{\frac{1}{48n \log n}, \frac{1}{4}\right\} \cdot \delta \cdot \text{vol}_{n-1}(f) = \frac{1}{48n \log n} \cdot \delta \cdot \text{vol}_{n-1}(f). \tag{3}$$

Since every such neighboring cube can be counted at most $2n$ times using this argument, we get at most $1/2n$ of the above volume in S_3. But f_2 might not be (fully) contained in K. So, we need to move to K'. Let $\mathcal{C}_2' = \{c \in \mathcal{C}_2 : c \cap K' \neq \emptyset\}$. Our choice of α ensures that the cubes in $\mathcal{C}_2'$ and all their neighboring cubes are fully contained in K. Let $\partial_{K'}(\mathcal{C}_2')$ be the boundary of $\mathcal{C}_2'$ relative to K'. Then $\partial_{K'}(\mathcal{C}_2') \subseteq \partial_K(\mathcal{C}_2)$ as $\partial_{K'}(\mathcal{C}_2')$ only consists of the boundary of $\mathcal{C}_2 \cap K'$ internal to K'. So, every facet $f \in \partial_{K'}(\mathcal{C}_2')$ contributes at least

$$\frac{1}{2n} \cdot \frac{1}{48n \log n} \cdot \delta \cdot \text{vol}_{n-1}(f)$$

to $\text{vol}(S_3)$. Because S_1 occupies at least $2/3$ of every cube in $\mathcal{C}_2'$,

$$\text{vol}(\mathcal{C}_2' \cap S_1) \leq \text{vol}(\mathcal{C}_2') \leq \frac{3}{2}\text{vol}(S_1 \cap K') \leq \frac{3}{4}\text{vol}(K') \tag{4}$$

and by Lemma 11,

$$\begin{aligned}
\mathrm{vol}\left(\partial_{K'}\left(\mathcal{C}_2'\right)\right) &\geq \frac{\ln 2}{R} \cdot \min\left\{\mathrm{vol}\left(\mathcal{C}_2'\right), \mathrm{vol}\left(K' \backslash \mathcal{C}_2'\right)\right\} \\
&\geq \frac{\ln 2}{R} \cdot \min\left\{\mathrm{vol}\left(\mathcal{C}_2 \cap K'\right), \frac{1}{4}\mathrm{vol}\left(K'\right)\right\} && \text{(from eq. (4))} \\
&\geq \frac{\ln 2}{R} \cdot \min\left\{\mathrm{vol}\left(\mathcal{C}_2 \cap K'\right), \frac{1}{3}\mathrm{vol}\left(\mathcal{C}_2 \cap K'\right)\right\} \\
&\geq \frac{\ln 2}{3R} \cdot \mathrm{vol}\left(\mathcal{C}_2 \cap K'\right). && (5)
\end{aligned}$$

Combining equation (3) and equation (5), we get

$$\begin{aligned}
\mathrm{vol}(S_3) &\geq \frac{1}{2n} \cdot \sum_{f \in \partial_{K'}(\mathcal{C}_2')} \frac{\delta}{48n \log n} \mathrm{vol}_{n-1}(f) \\
&= \frac{\delta}{96n^2 \log n} \cdot \mathrm{vol}(\partial_{K'}(\mathcal{C}_2')) \geq \frac{\delta \ln 2}{300Rn^2 \log n} \cdot \mathrm{vol}\left(\mathcal{C}_2 \cap K'\right) \\
&\geq \frac{\delta \ln 2}{300Rn^2 \log n} \cdot \mathrm{vol}\left(\mathcal{C}_2 \cap K' \cap S_1\right) \\
&\geq \frac{\delta \ln 2}{300Rn^2 \log n} \cdot \left(\mathrm{vol}\left(\mathcal{C}_2 \cap S_1\right) - (1-(1-\alpha)^n)\mathrm{vol}(K)\right) \\
&\geq \frac{\delta \ln 2}{300Rn^2 \log n} \cdot \left(\frac{1}{2}\mathrm{vol}(S_1) - \frac{\epsilon}{2}\mathrm{vol}(K)\right) \\
&\geq \frac{\delta \ln 2}{600Rn^2 \log n} \cdot \left(\mathrm{vol}(S_1) - \epsilon \mathrm{vol}(K)\right)
\end{aligned}$$

Using $\delta = \frac{\alpha}{4\sqrt{n}} = \frac{\epsilon}{80n\sqrt{n}}$, we have

$$\mathrm{vol}(S_3) \geq \frac{\epsilon \ln 2}{48 \cdot 10^3 \cdot Rn^{3.5} \log n} \cdot \left(\mathrm{vol}(S_1) - \epsilon \mathrm{vol}(K)\right). \qquad \blacktriangleleft$$

3 Conductance

Here we bound the s-conductance of CHAR. The following simple lemma lets us reduce the s-conductance of K to isoperimetry of axis-disjoint subsets in K.

▶ **Lemma 12.** *Let $S_1 \subseteq K$ be a measurable subset of K and $S_2 = K \backslash S_1$. Let $S_1' = \{x \in S_1 : P_x(S_2) < \frac{1}{2n}\}$ and $S_2' = \{x \in S_2 : P_x(S_1) < \frac{1}{2n}\}$. Then S_1' and S_2' are axis disjoint.*

Proof. Assume S_1' and S_2' are not axis-disjoint, then let ℓ be an axis-parallel line passing through both S_1' and S_2'. Let $x \in S_1' \cap \ell$ and $y \in S_2' \cap \ell$. Then

$$P_x(S_2) \geq \frac{1}{n}\frac{\mathrm{len}(\ell \cap S_2)}{\mathrm{len}(\ell \cap K)} \Rightarrow \mathrm{len}(\ell \cap S_2) < \frac{\mathrm{len}(\ell \cap K)}{2}$$

and

$$P_y(S_1) \geq \frac{1}{n}\frac{\mathrm{len}(\ell \cap S_1)}{\mathrm{len}(\ell \cap K)} \Rightarrow \mathrm{len}(\ell \cap S_1) < \frac{\mathrm{len}(\ell \cap K)}{2}.$$

This is a contradiction as $\mathrm{len}(\ell \cap K) = \mathrm{len}(\ell \cap S_1) + \mathrm{len}(\ell \cap S_2)$. $\blacktriangleleft$

▶ **Theorem 13.** *Let K be a convex body in $\mathbb{R}^n$ containing a unit ball with $R^2 = \mathbb{E}_K(\|x - z_K\|^2)$ where z_K is the centroid of K. Then the s-conductance of coordinate hit-and-run in K is at least $\frac{s}{8 \cdot 10^5 \cdot Rn^{4.5} \log n}$.*

Proof. Let $S_1 \subseteq K$ be a measurable subset of K with $s < \pi_K(S_1) \leq 1/2$ and let $S_2 = K \backslash S_1$. Let

$$S_1' = \{x \in S_1 : P_x(S_2) < \frac{1}{2n}\} \quad \text{and} \quad S_2' = \{x \in S_2 : P_x(S_1) < \frac{1}{2n}\}.$$

Let $S_3' = K \backslash \{S_1' \cup S_2'\}$. From Lemma 12, we know that S_1' and S_2' are axis-disjoint. Thus, from Theorem 3 with $\epsilon = s/2$, we get

$$\text{vol}(S_3') \geq \frac{s}{96 \cdot 10^3 \cdot Rn^{3.5} \log n} \left(\min\{\text{vol}(S_1'), \text{vol}(S_2')\} - \frac{s}{2}\text{vol}(K)\right).$$

If $\text{vol}(S_1') < \text{vol}(S_1)/2$, then

$$\begin{aligned}\int_{x \in S_1} P_x(S_2)dx &= \int_{x \in S_1'} P_x(S_2)dx + \int_{x \in S_1 \backslash S_1'} P_x(S_2)dx \\ &\geq \frac{1}{2n}\text{vol}(S_1 \backslash S_1') \geq \frac{1}{4n}\text{vol}(S_1),\end{aligned}$$

and $\phi_s(S_1) \geq \frac{1}{4n}$. If $\text{vol}(S_2') < \text{vol}(S_2)/2$, then

$$\int_{x \in S_1} P_x(S_2)dx = \int_{x \in S_2} P_x(S_1)dx \geq \frac{1}{4n}\text{vol}(S_2).$$

So, assume that $\text{vol}(S_1') \geq \text{vol}(S_1)/2$ and $\text{vol}(S_2') \geq \text{vol}(S_2)/2$. Then,

$$\int_{x \in S_1} P_x(S_2)dx \geq \int_{x \in S_1 \backslash S_1'} P_x(S_2)dx \geq \frac{1}{2n} \cdot \text{vol}(S_1 \backslash S_1'), \tag{6}$$

and

$$\int_{x \in S_1} P_x(S_2)dx \geq \int_{x \in S_1} P_x(S_2 \backslash S_2')dx = \int_{y \in S_2 \backslash S_2'} P_y(S_1)dy \geq \frac{1}{2n}\text{vol}(S_2 \backslash S_2'). \tag{7}$$

Thus, from equations (6) and (7),

$$\begin{aligned}\int_{x \in S_1} P_x(S_2)dx &\geq \frac{1}{2} \cdot \frac{1}{2n} \cdot (\text{vol}(S_1 \backslash S_1') + \text{vol}(S_2 \backslash S_2')) = \frac{1}{4n} \cdot \text{vol}(S_3') \\ &\geq \frac{s}{400 \cdot 10^3 \cdot Rn^{4.5} \log n} \cdot \left(\min\{\text{vol}(S_1'), \text{vol}(S_2')\} - \frac{s}{2} \cdot \text{vol}(K)\right) \\ &\geq \frac{s}{800 \cdot 10^3 \cdot Rn^{4.5} \log n} (\min\{\text{vol}(S_1), \text{vol}(S_2)\} - s \cdot \text{vol}(K)) \\ &\geq \frac{s}{8 \cdot 10^5 \cdot Rn^{4.5} \log n} (\text{vol}(S_1) - s \cdot \text{vol}(K)) \\ &= \frac{\text{vol}(K)s}{8 \cdot 10^5 \cdot Rn^{4.5} \log n} (\pi_K(S_1) - s) \\ \Rightarrow \int_{S_1} P_x(K \backslash S_1)d\pi_K(x) &\geq \frac{s}{8 \cdot 10^5 \cdot Rn^{4.5} \log n} (\pi_K(S_1) - s).\end{aligned}$$

Thus, for any $S_1 \subseteq K$ with $s < \pi_K(S_1) \leq 1/2$, we get

$$\frac{p(S_1)}{\pi_K(S_1) - s} \geq \frac{\int_{x \in S_1} P_x(K \backslash S_1)d\pi_K(x)}{\pi_K(S_1) - s} \geq \frac{s}{8 \cdot 10^5 \cdot Rn^{4.5} \log n},$$

which implies that the s-conductance of the CHAR Markov chain is

$$\phi_s \geq \frac{s}{8 \cdot 10^5 \cdot Rn^{4.5} \log n}.$$

◀

Proof of Theorem 1. For a convex body K, let π_0 be the starting distribution and π_t be the distribution after t steps of CHAR. Assume that π_0 is M-warm with respect to π_K. From Theorem 4 and selecting $s = \frac{\epsilon}{2M}$, we get

$$d_{TV}(\pi_t, \pi_K) \leq \frac{\epsilon}{2} + M\left(1 - \frac{\phi_s^2}{2}\right)^t.$$

Combining this with Theorem 13,

$$t = \frac{4}{\phi_s^2} \log \frac{2M}{\epsilon} = O\left(\frac{M^2 R^2 n^9 \log^2 n}{\epsilon^2} \log \frac{2M}{\epsilon}\right)$$

steps of the CHAR Markov chain suffice to ensure $d_{TV}(\pi_t, \pi_K) \leq \epsilon$. ◀

4 Lower bound

▶ **Theorem 14.** *For a convex body K in $\mathbb{R}^n$ with diameter D and containing a unit ball, the conductance of Coordinate Hit-and-Run is $O(1/n^2 D)$.*

Proof. Fix a simplex C in $\mathbb{R}^{n-1}$ with center of gravity at zero containing a unit ball. We construct a convex body K in $\mathbb{R}^n$ so that $K(x_1)$, the slice of K with the first coordinate x_1, is $C + (x_1, 0, \ldots, 0)$ for $x_1 \in [0, D]$ and empty outside this range of x_1. We choose $D \geq 2n$. Let $S \subset K$ be the set of all points in K with $x_1 \leq D/2$. We now observe that the volume of axis-aligned extension of S, $\text{ext}_K(S)$ is bounded by $O(1/nD)$ times the volume of S. To see this, note that the shadow of the cross-section has volume that can be computed as

$$\int_0^h t^{n-2}(h-t) = h^{n-1}\left(\frac{1}{n-1} - \frac{1}{n}\right) = \frac{h^{n-1}}{n(n-1)} = \frac{A}{n}$$

where A is the area of the cross-section. This shows that the isoperimetric ratio is $O(1/nD)$. Next, we note that the extension of S goes beyond S only along e_1, and the probability that CHAR chooses e_1 at any step is only $1/n$. This gives a conductance bound of $O(1/(n^2 D))$. ◀

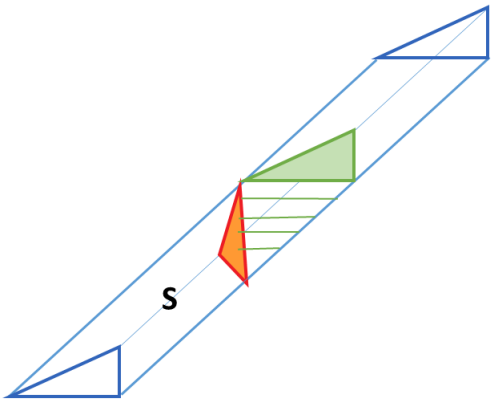


Figure 3 The lower bound construction.

We expect that this translates to a lower bound of $\widetilde{\Omega}(n^3D^2)$ on the mixing rate even from a warm start. Consider two subsets of K at opposite ends: $K \cap \{x : x_1 \leq D/4\}$ and $K \cap \{x : x_1 \geq 3D/4\}$. Suppose we start with a uniformly random point in the first set. Then in order to mix, the current point must reach the latter set. Even though this is worse than the $\widetilde{O}(n^2D^2)$ mixing rate of hit-and-run, it is an interesting open problem to determine the precise mixing rate of CHAR.

References

1. Hans C. Andersen and Persi Diaconis. Hit and run as a unifying device. *Journal de la société française de statistique*, 148(4):5–28, 2007. URL: `http://www.numdam.org/item/JSFS_2007__148_4_5_0`.
2. Arnon Boneh. Preduce – a probabilistic algorithm identifying redundancy by a random feasible point generator (rfpg). In *Redundancy in Mathematical Programming*, pages 108–134, Berlin, Heidelberg, 1983. Springer Berlin Heidelberg.
3. Ben Cousins and Santosh Vempala. Volume computation of convex bodies. *MATLAB File Exchange*, 2013. URL: `http://www.mathworks.com/matlabcentral/fileexchange/43596-volume-computation-of-convex-bodies`.
4. Ben Cousins and Santosh Vempala. A practical volume algorithm. *Mathematical Programming Computation*, 8(2):133–160, 2016.
5. Persi Diaconis, Kshitij Khare, and Laurent Saloff-Coste. Gibbs sampling, conjugate priors and coupling. *Sankhya A*, 72(1):136–169, 2010.
6. Persi Diaconis, Gilles Lebeau, and Laurent Michel. Gibbs/metropolis algorithms on a convex polytope. *Mathematische Zeitschrift*, 272(1-2):109–129, 2012.
7. Ioannis Z Emiris and Vissarion Fisikopoulos. Efficient random-walk methods for approximating polytope volume. In *Proceedings of the thirtieth annual symposium on Computational geometry*, pages 318–327, 2014.
8. Jenny Rose Finkel, Trond Grenager, and Christopher D Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 363–370, 2005.
9. Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 6(6):721–741, 1984. `doi:10.1109/TPAMI.1984.4767596`.
10. Edward I George and Robert E McCulloch. Variable selection via gibbs sampling. *Journal of the American Statistical Association*, 88(423):881–889, 1993.
11. R. Kannan, L. Lovász, and M. Simonovits. Isoperimetric problems for convex bodies and a localization lemma. *Discrete & Computational Geometry*, 13:541–559, 1995.
12. R. Kannan, L. Lovász, and M. Simonovits. Random walks and an $O^*(n^5)$ volume algorithm for convex bodies. *Random Structures and Algorithms*, 11:1–50, 1997.
13. Yin Tat Lee and Santosh Srinivas Vempala. Eldan's stochastic localization and the KLS hyperplane conjecture: An improved lower bound for expansion. In *Proc. of IEEE FOCS*, 2017.
14. L. Lovász. How to compute the volume? *Jber. d. Dt. Math.-Verein, Jubiläumstagung 1990*, pages 138–151, 1990.
15. L. Lovász. Hit-and-run mixes fast. *Math. Prog.*, 86:443–461, 1998.
16. L. Lovász and M. Simonovits. Random walks in a convex body and an improved volume algorithm. In *Random Structures and Alg.*, volume 4, pages 359–412, 1993.
17. L. Lovász and S. Vempala. Hit-and-run from a corner. *SIAM J. Computing*, 35:985–1005, 2006.
18. L. Lovász and S. Vempala. The geometry of logconcave functions and sampling algorithms. *Random Struct. Algorithms*, 30(3):307–358, 2007. `doi:10.1002/rsa.v30:3`.

19 László Lovász and Miklós Simonovits. Random walks in a convex body and an improved volume algorithm. *Random structures & algorithms*, 4(4):359–412, 1993.

20 Hariharan Narayanan and Piyush Srivastava. On the mixing time of coordinate hit-and-run. *arXiv preprint*, 2020. arXiv:2009.14004.

21 A. Sinclair and M. Jerrum. Approximate counting, uniform generation and rapidly mixing Markov chains. *Information and Computation*, 82:93–133, 1989.

22 R.L. Smith. Efficient Monte-Carlo procedures for generating points uniformly distributed over bounded regions. *Operations Res.*, 32:1296–1308, 1984.

23 V. Turchin. On the computation of multidimensional integrals by the monte-carlo method. *Theory of Probability & Its Applications*, 16(4):720–724, 1971. doi:10.1137/1116083.

Combinatorial Resultants in the Algebraic Rigidity Matroid

Goran Malić
Computer Science Department, Smith College, Northampton, MA, USA

Ileana Streinu[1]
Computer Science Department, Smith College, Northampton, MA, USA

Abstract

Motivated by a rigidity-theoretic perspective on the *Localization Problem* in 2D, we develop an algorithm for computing *circuit polynomials* in the algebraic rigidity matroid CM_n associated to the Cayley-Menger ideal for n points in 2D. We introduce *combinatorial resultants*, a new operation on graphs that captures properties of the Sylvester resultant of two polynomials in the algebraic rigidity matroid. We show that every rigidity circuit has a *construction tree* from K_4 graphs based on this operation. Our algorithm performs an *algebraic elimination* guided by the construction tree, and uses classical resultants, factorization and ideal membership. To demonstrate its effectiveness, we implemented our algorithm in Mathematica: it took less than 15 seconds on an example where a Gröbner Basis calculation took 5 days and 6 hrs.

2012 ACM Subject Classification General and reference → Performance; General and reference → Experimentation; Theory of computation → Computational geometry; Mathematics of computing → Matroids and greedoids; Mathematics of computing → Mathematical software performance; Computing methodologies → Combinatorial algorithms; Computing methodologies → Algebraic algorithms

Keywords and phrases Cayley-Menger ideal, rigidity matroid, circuit polynomial, combinatorial resultant, inductive construction, Gröbner basis elimination

Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.52

Related Version *Full Version*: `https://arxiv.org/abs/2103.08432` [21]

Supplementary Material Polynomials computed by our algorithm are made available in Mathematica's compressed and portable wdx format at the following GitHub repository:
Dataset: `https://github.com/circuitPolys/CayleyMenger`
archived at `swh:1:dir:43ae808290f56bde92a4139cf1b72f4f2f57adf8`

Funding Both authors acknowledge funding from the NSF CCF:1703765 grant to Ileana Streinu.

1 Introduction

This paper addresses combinatorial, algebraic and algorithmic aspects of a question motivated by the following ubiquitous problem from *distance geometry*:

Localization. A graph together with *weights* associated to its edges is given. The goal is to find *placements* of the graph in some Euclidean space, so that the edge lengths match the given weights. In this paper, we work in 2D. A system of quadratic equations can be easily set up so that the possible placements are among the (real) solutions of this system. Rigidity Theory can help predict, *a priori,* whether the set of solutions will be discrete (if the

[1] Corresponding author

given weighted graph is *rigid*) or continuous (if the graph is *flexible*). In the rigid case, the double-exponential Gröbner basis algorithm can be used, *in principle,* to eliminate all but one of the variables. Once a polynomial in a single variable is obtained, numerical methods are used to solve it. We then select one solution, substitute it in the original equations, eliminate to get a polynomial in a new variable and repeat.

Single unknown distance problem. Instead of attempting to directly compute the coordinates of all the vertices, we restrict our attention to the related problem of finding the possible values of a *single unknown distance* corresponding to a *non-edge* (a pair of vertices that are not connected by an edge). Indeed, *if* we could solve this problem for a collection of non-edge pairs that form a trilateration when added to the original collection of edges, *then* a single solution in Cartesian coordinates could be easily computed afterwards in linearly many steps of quadratic equation solving.

Rigidity circuits. We formulate the *single unknown distance* problem in terms of Cayley-coordinates (squared distances between points) rather than Cartesian xy-coordinates. Known theorems from Distance Geometry, Rigidity Theory and Matroid Theory help reduce this problem to finding a certain irreducible polynomial in the Cayley-Menger ideal, called the *circuit polynomial*. Its support is a graph called a *circuit in the rigidity matroid*, or shortly a *rigidity circuit.* Substituting given edge lengths in the circuit polynomial results in a uni-variate polynomial which can be solved for the unknown distance.

The focus of this paper is the following:

▶ **Main Problem.** Given a rigidity circuit, compute its corresponding circuit polynomial.

Related work. While both *distance geometry* and *rigidity theory* have a distinguished history for which a comprehensive overview would be too long to include here, very little is known about computing *circuit polynomials. To the best of our knowledge,* their study in *arbitrary* polynomial ideals was initiated in the PhD thesis of Rosen [23]. His Macaulay2 code [24] is useful for exploring small cases, but the Cayley-Menger ideal is beyond its reach. A recent article [25] popularizes algebraic matroids and uses for illustration the smallest circuit polynomial K_4 in the Cayley-Menger ideal. *We could not find non-trivial examples anywhere.* Indirectly related to our problem are results such as [28], where an explicit univariate polynomial of degree 8 is computed (for an unknown angle in a $K_{3,3}$ configuration given by edge lengths, from which the placement of the vertices is determined) and [26], for its usage of Cayley coordinates in the study of configuration spaces of some families of distance graphs. A closely related problem is that of computing the *number of embeddings of a minimally rigid graph* [4], which has received *a lot* of attention in recent years (e.g. [6, 1, 10, 9], to name a few). References to specific results in the literature that are relevant to the theory developed here and to our proofs are given throughout the paper.

How tractable is the problem? Circuit polynomial computations can be done, in principle, with the double-exponential time Gröbner basis algorithm with an elimination order. The largest we could do with the **GroebnerBasis** function of Mathematica 12 (running on a 2019 iMac computer with 6 cores at 3.7Ghz and 16GB RAM) was the Desargues-plus-one (Fig. 1) circuit ($658,175$ terms), which took 5 days and 6 hours. In all other cases the execution timed out or crashed. Our goal is to make such calculations *more tractable* by taking advantage of *structural information* inherent in the problem.

Our results. We describe a new *algorithm to compute a circuit polynomial with known support.* It relies on resultant-based elimination steps guided by a novel *inductive construction for rigidity circuits.* While inductive constructions have been often used in Rigidity Theory, most notably the Henneberg sequences for Laman graphs [14] and Henneberg II sequences for rigidity circuits [2], we argue that our construction is more *natural* due to its *direct algebraic interpretation.* We have implemented our method in Mathematica and applied it successfully to compute all circuit polynomials on up to 6 vertices and a few on 7 vertices, the largest of which having over two million terms. The previously mentioned example that took over 5 days to complete with GroebnerBasis, was solved by our algorithm in less than 15 seconds.

Main theorems. We first define the *combinatorial resultant* of two graphs as an abstraction of the classical resultant. Our main theoretical result is split into the combinatorial Theorem 1 and the algebraic Theorem 2, each with an algorithmic counterpart.

▶ **Theorem 1.** *Each rigidity circuit can be obtained, inductively, by applying combinatorial resultant operations starting from K_4 circuits. The construction is captured by a binary* resultant tree *whose nodes are intermediate rigidity circuits and whose leaves are K_4 graphs.*

Theorem 1 leads to a *graph algorithm* for finding a *combinatorial resultant tree* of a circuit. Each step of the construction can be carried out in polynomial time using variations on the *Pebble Game* matroidal sparsity algorithms [18] combined with Hopcroft and Tarjan's linear time 3-connectivity algorithm [15]. However, it is conceivable that the resultant tree could be exponentially large with non-repeating subtrees, and thus the entire construction could take an exponential number of steps: understanding in detail the algorithmic complexity of our method *remains a problem for further investigation.*

▶ **Theorem 2.** *Each circuit polynomial can be obtained, inductively, from K_4 circuit polynomials by applying resultant operations in a manner guided by the combinatorial resultant tree from Theorem 1. At each step, the resultant produces a polynomial that may not be irreducible. A polynomial factorization and a test of membership in the ideal may need to be applied to identify the factor which is the circuit polynomial.*

The resulting *algebraic elimination algorithm* runs in exponential time, in part because of the growth in size of the polynomials that are being produced. Several interesting theoretical *open questions* remain, whose answers may affect the precise time complexity analysis.

Computational experiments. We implemented our algorithms in Mathematica V12.1.1.0 on two computers with the following specifications: Intel i5-9300H 2.4GHz, 32 GB RAM, Windows 10 64-bit; and Intel i5-9600K 3.7GHz, 16 GB RAM, macOS Mojave 10.14.5. We also explored Macaulay2, but it was much slower than Mathematica (hours vs. seconds) in computing one of our examples. The polynomials resulting from our calculations, summarized in Table 1 at the end of the paper, are made available on a github repository [19].

Overview of the paper. In Section 2 we introduce the background concepts from matroid theory and rigidity theory. We introduce combinatorial resultants in Section 3, prove Theorem 1 and describe the algorithm for computing a combinatorial resultant tree. In Section 4 we introduce the background concepts pertaining to algebraic matroids in the Cayley-Menger ideal and elimination theory. In Section 5 we prove Theorem 2. We conclude in Section 6 with a summary of the preliminary experimental results we carried with our implementation. For complete details and a self-contained presentation, including relevant

definitions and results from Rigidity Theory, Matroid Theory, Ideals, the Cayley-Menger ideal, Resultants and Elimination theory, complete proofs and further details on our results, the reader should refer to the full version of the paper, currently available on the arxiv [21].

2 Preliminaries: circuits in the rigidity matroid

We start with the combinatorial aspects of our problem. In this section we review well-known concepts and results from combinatorial rigidity theory that are relevant for our paper.

Notation. We work with (sub)graphs given by subsets E of edges of the complete graph K_n on vertices $[n] := \{1, \cdots, n\}$. If G is a (sub)graph, then $V(G)$, resp. $E(G)$ denote its vertex, resp. edge set. The support of G is $E(G)$. The *vertex span* $V(E)$ of edges E is the set of all edge-endpoint vertices. A subgraph G is *spanning* if its edge set $E(G)$ spans $[n]$. The *neighbours* $N(v)$ of v are the vertices adjacent to v in G.

Rigid graphs. Let $G = (V, E)$ be an undirected graph and let $\ell = \{\ell_{ij} | ij \in E\}$ be a collection of numbers interpreted as *lengths* associated to its edges. Up to rigid transformations, *in how many ways can we place the vertices of G at points in the plane, so that the segments corresponding to edges have the prescribed lengths?* The answer can be a finite number or a continuum of possibilities, and it depends in principle on both G and ℓ. For all but a measure zero set of possible lengths (said to be *generic*[2]), the answer depends only on G. We say that G is *rigid* if, for generic edge lengths, the number of placements is finite, and *flexible* otherwise. G is said to be *minimally rigid* if it is rigid, but it becomes flexible when any of its edges is removed.

Laman graphs. Minimally rigid graphs are characterized by Laman's Theorem [17]: a graph $G = (V, E)$ is minimally rigid in 2D iff (a) G is $(2, 3)$-sparse (no subset of $n' \leq n = |V|$ vertices spans more than $2n' - 3$ edges) and (b) G is $(2, 3)$-tight (has a total of $2n - 3$ edges). Graphs with these two properties will be called *Laman graphs*. A *Laman-plus-one graph* is obtained by adding one edge to a Laman graph: it has a total of $2n - 2$ edges and thus it violates the $(2, 3)$-sparsity condition on V and possibly on some other proper subsets of V.

Matroids. A matroid is an abstraction capturing (in)dependence relations among collections of elements from a *ground set*, and is inspired by both *linear* dependencies (among, say, rows of a matrix) and by *algebraic* constraints imposed by algebraic equations on a collection of otherwise free ("independent") variables. The standard way to specify a matroid is via its *independent sets*, which have to satisfy certain axioms [22] (skipped here, since they are not relevant for our presentation). A *base* is a maximal independent set and a *dependent* set is one which is not independent. A *minimal dependent set* is called a *circuit*. Relevant for our purposes are the following general aspects: (a) (hereditary property) a subset of an independent set is also independent; (b) all bases have the same cardinality, called the *rank* of the matroid. Further properties will be introduced in context, as needed. In this paper we encounter two types of matroids: a *graphic matroid*, defined on a ground set given by all the edges $E_n := \{ij : 1 \leq i < j \leq n\}$ of the complete graph K_n; this is the $(2, 3)$-*sparsity*

[2] Many results in Rigidity Theory are often using the very strong assumption that the edge lengths should be *algebraically independent* in order to be *generic*; this assumption simplifies the proofs but is not necessary and would preclude any effective algorithmic application of generic graphs.

matroid or the *generic rigidity matroid* described below; and an *algebraic matroid*, defined on an isomorphic ground set of variables $X_n := \{x_{ij} : 1 \leq i < j \leq n\}$; this is the *algebraic matroid for the Cayley-Menger ideal* and will be introduced in Section 4. These matroids are isomorphic (for a comprehensive proof, see [21]).

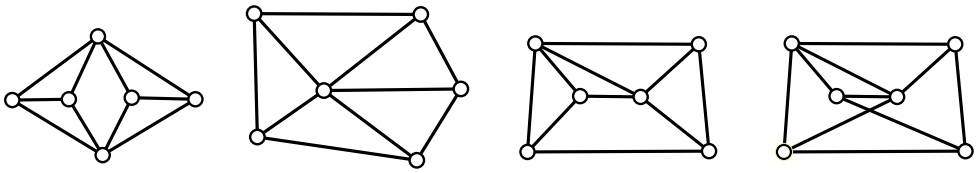


Figure 1 The four types of circuits on $n = 6$ vertices: 2D *double-banana*, 5-*wheel* W_5, *Desargues-plus-one* and $K_{3,3}$-plus-one.

Circuits in the Rigidity Matroid. Laman graphs form the *bases* of the so-called (generic) *2D rigidity matroid*. The *independent sets* are the $(2, 3)$-sparse graphs; the *dependent* sets violate the $(2, 3)$-sparsity condition on at least one subset of vertices. A *(rigidity) circuit* is a dependent graph with minimum edge support: removing any of its edges leads to a Laman graph on its spanned vertex set. A *spanning rigidity circuit* has V as its vertex set (Fig. 1). A *Laman-plus-one* graph contains a unique circuit. A spanning rigidity circuit $C = (V, E)$ is a special case of a Laman-plus-one graph: it has a total of $2n - 2$ edges and it satisfies the $(2, 3)$-sparsity condition on all proper subsets of $n' \leq n - 1$ vertices.

Operations on circuits. If G_1 and G_2 are two graphs, we use a consistent notation for their number of vertices and edges $n_i = |V(G_i)|$, $m_i = |E(G_i)|$, $i = 1, 2$, and for their union and intersection of vertices and edges, as in $V_\cup = V(G_1) \cup V(G_2)$, $V_\cap = V(G_1) \cap V(G_2)$, $n_\cup = |V_\cup|$, $n_\cap = |V_\cap|$ and similarly for edges, with $m_\cup = |E_\cup|$ and $m_\cap = |E_\cap|$. Let C_1 and C_2 be two circuits with exactly one common edge uv. Their 2-*sum* $C := C_1 \otimes C_2$ is the graph $C = (V_\cup, E_\cup \setminus \{uv\})$. The inverse operation of splitting C into C_1 and C_2 is called a 2-split [3] (Fig. 2). It is easy to see that the 2-sum of two circuits is a circuit, and that any 2-split of a circuit gives a pair of circuits ([2], Lemmas 4.1, 4.2 or use sparsity).

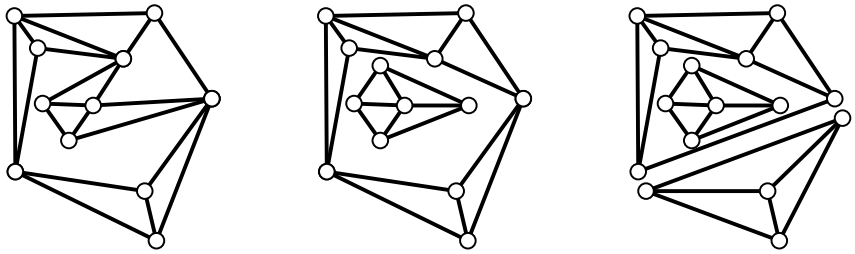


Figure 2 Splitting twice a 2-connected circuit (left) to get three 3-connected circuits (right).

[3] The 2-sum operation assumes that the two vertices of the summands coincide, and the 2-split operation produces two graphs, shown as disjoint in Fig. 2, for clarity.

Connectivity. A circuit is always a 2-connected graph (this follows easily from sparsity), and for simplicity we refer to one that is *not* 3-*connected* as a 2-*connected circuit*. The Tutte decomposition [27] of a graph into 3-connected components, applied on a circuit, induces 2-splits and produces smaller circuits: any 2-connected circuit can be constructed from smaller 3-connected circuits via 2-sums (Fig. 2).

Inductive constructions for 3-connected circuits. A *Henneberg II* extension (also called an *edge splitting* operation) is defined for an edge uv and a non-incident vertex w, as follows: the edge uv is removed, a new vertex a and three new edges au, av, aw are added. Berg and Jordan [2] have shown that, if G is a 3-connected circuit, then a Henneberg II extension on G is also a 3-connected circuit. The *inverse Henneberg II* operation on a circuit removes one vertex of degree 3 and adds a new edge among its three neighbors in such a way that the result is also a circuit. Berg and Jordan have shown that every 3-connected circuit admits an inverse Henneberg II operation which also maintains 3-connectivity. As a consequence, a 3-connected circuit has an *inductive construction*, i.e. it can be obtained from a single K_4 by Henneberg II extensions that maintain 3-connectivity. Their proof is based on the existence of two non-adjacent vertices with 3-connected inverse Henneberg II circuits. We will make use in Section 3 of the following weaker result, which does not require the maintenance of 3-connectivity in the inverse Henneberg II operation.

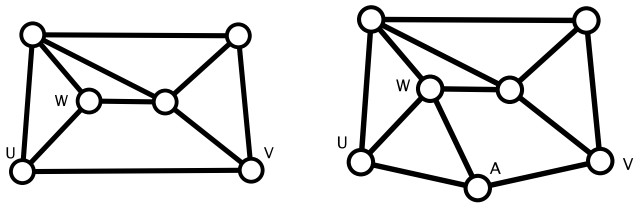


Figure 3 A Henneberg II extension of the Desargues-plus-one circuit.

▶ **Lemma 3** (Theorem 3.8 in [2]). *A 3-connected circuit with at least 5 vertices has either (a) four vertices that admit an inverse Henneberg II to a circuit, or (b) three pairwise non-adjacent vertices that admit an inverse Henneberg II to a circuit (not necessarily 3-connected).*

3 Combinatorial resultants

We define now the *combinatorial resultant* operation on two graphs, prove Theorem 1 and describe its algorithmic implications.

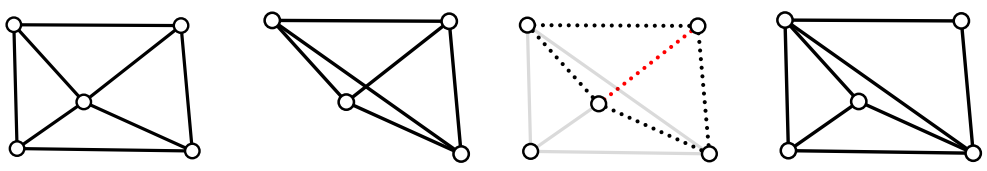


Figure 4 Circuit-invalid combinatorial resultant of two properly intersecting circuits. Left to right: a 4-wheel W_4, a complete K_4 graph, their common Laman graph (dotted, with red elimination edge) and the combinatorial resultant, which is a non-circuit Laman-plus-one graph.

Combinatorial resultant. Let G_1 and G_2 be two distinct graphs with non-empty intersection $E_\cap \neq \emptyset$ and let $e \in E_\cap$ be a common edge. The *combinatorial resultant* of G_1 and G_2 on the *elimination edge* e is the graph $\mathrm{CRes}(G_1, G_2, e)$ with vertex set $V_\cup$ and edge set $E_\cup \setminus \{e\}$.

The 2-sum is the special case when the two graphs have exactly one edge in common. Circuits are closed under the 2-sum operation, but they are not closed under general combinatorial resultants (Fig. 4). We are interested in combinatorial resultants that produce circuits from circuits. The following Lemma gives a necessary condition.

▶ **Lemma 4.** *The combinatorial resultant of two circuits has $m = 2n-2$ edges iff the common subgraph $G_\cap$ of the two circuits is Laman.*

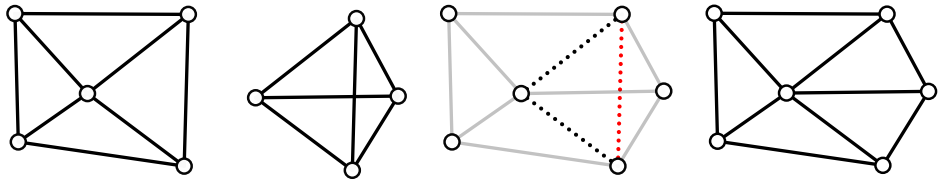


Figure 5 Circuit-valid combinatorial resultant of two properly intersecting circuits. Left to right: a 4-wheel W_4, a complete K_4 graph, their common Laman graph (dotted, with red elimination edge) and their combinatorial resultant, the 5-wheel W_5 circuit.

Proof. Let C be the combinatorial resultant with n vertices and m edges of two circuits C_1 and C_2 with n_i vertices and m_i edges, $i = 1, 2$. By inclusion-exclusion $n = n_1 + n_2 - n_\cap$ and $m = m_1 + m_2 - m_\cap - 1$. Substituting here the values for $m_1 = 2n_1 - 2$ and $m_2 = 2n_2 - 2$, we get $m = 2n_1 - 2 + 2n_2 - 2 - m_\cap - 1 = 2(n_1 + n_2 - n_\cap) - 2 + 2n_\cap - 3 - m_\cap = (2n-2) + (2n_\cap - 3) - m_\cap$. We have $m = 2n - 2$ iff $m_\cap = 2n_\cap - 3$. Since both C_1 and C_2 are circuits, it is not possible that one edge set be included in the other: circuits are minimally dependent sets of edges and thus cannot contain other circuits. As a proper subset of both $E_1 = E(C_1)$ and $E_2 = E(C_2)$, $E_\cap$ satisfies the hereditary $(2, 3)$-sparsity property. If furthermore $G_\cap$ has exactly $2n_\cap - 3$ edges, then it is Laman. ◀

Circuit-valid combinatorial resultant sequences. Two circuits are said to be *properly intersecting* if their common subgraph is Laman. Being properly intersecting is a necessary but not sufficient condition for the combinatorial resultant of two circuits to produce a circuit (Fig. 4). A combinatorial resultant operation applied to two properly intersecting circuits is said to be *circuit-valid* if it results in a spanning circuit (Fig. 5).

▶ **Open Problem 1.** *Find necessary and sufficient conditions for the combinatorial resultant of two circuits to be a circuit.*

In Section 2 we have seen that a 2-connected circuit can be obtained from 3-connected circuits via 2-sums. To complete the proof of Theorem 1 we show now:

▶ **Proposition 5.** *Let $C = (V, E)$ be a 3-connected circuit spanning $n + 1 \geq 5$ vertices. Then, in polynomial time, we can find two circuits A and B such that A has n vertices, B has at most n vertices and C is a circuit-valid combinatorial resultant of A and B. Algorithm 1 summarizes the procedure.*

Algorithm 1 Inverse Combinatorial Resultant.

Input: 3-connected circuit C
Output: circuits A, B and edge e such that $C = \mathrm{CRes}(A, B, e)$

```
for each vertex a of degree 3 do
    if inverse Henneberg II is possible on a
    and there is a non-adjacent degree 3 vertex b then
        Get circuit A and edge e by inverse Henneberg II in C on a
        Let D = C without b (and its edges) and with new edge e
        Compute unique circuit B in D
        return circuits A, B and edge e
    end if
end for
```

Proof. We apply a weaker version of Lemma 3 to find two non-adjacent vertices a and b of degree 3 such that a circuit A can be produced via an inverse Henneberg II operation on vertex a in C (see Fig. 6). Let the neighbors $N(a) = \{u, v, w\}$ of vertex a be labeled such that $e = uw$ is a non-edge in C and becomes an edge in the new circuit $A = (V \setminus \{a\}, E \setminus \{au, av, aw\} \cup \{uv\})$.

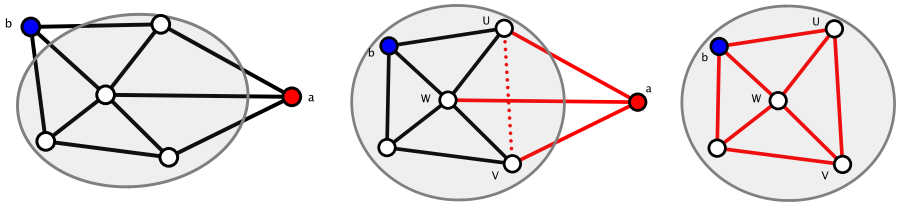


Figure 6 The 3-connected circuit C spanning $n + 1$ vertices with two non-adjacent vertices a (red) and b (blue) of degree 3. Their neighbors $N(a)$ and $N(b)$ may not be disjoint. An inverse Henneberg II at a removes the red edges at a and adds dotted red edge $e = uv$. Circuit A (red).

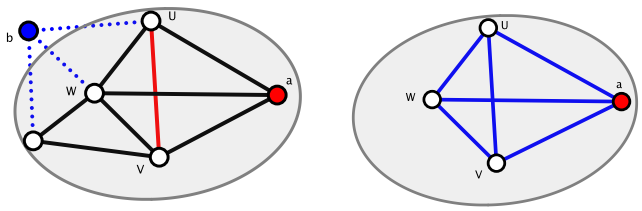


Figure 7 Remove from C the edges from b (blue dotted) and add red edge e. Circuit B (blue).

To define circuit B, we first let L be the subgraph of C induced by $V \setminus \{b\}$. Simple sparsity considerations show that L is a Laman graph. The graph D obtained from L by adding the edge $e = uv$, as in Fig. 7 (left), is a Laman-plus-one graph containing the three edges incident to a (which are not in A) and the edge e (which is in A). Thus D contains a unique circuit B (Fig. 7 left) with edge $e \in B$ (see e.g. [22, Proposition 1.1.6]). It remains to prove that B contains a and its three incident edges. If B does not contain a, then it is a proper subgraph of A: this contradicts the minimality of A as a circuit. Therefore a is a vertex in B, and because a vertex in a circuit can not have degree less than 3, it follows that B must contain all three of its incident edges.

The combinatorial resultant $\mathrm{CRes}(A, B, e)$ of the circuits A and B with e the eliminated edge satisfies the desired property that $C = \mathrm{CRes}(A, B, e)$. Algorithm 1 captures this procedure. The main steps, the Inverse Henneberg II step on a circuit at line 4 and finding the unique circuit in a Laman-plus-one graph at line 6 can be done in polynomial time using properties of the $(2, 3)$ and $(2, 2)$-sparsity pebble games from [18]. ◀

▶ **Corollary 6.** *The representation of C as the combinatorial resultant of two smaller circuits is, in general, not unique: an example is the 2D "double-banana" (Fig. 8).*

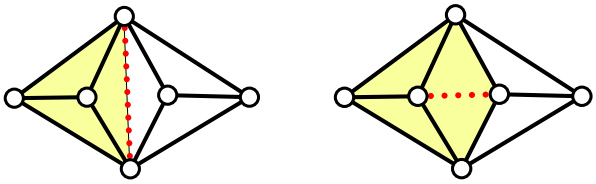


Figure 8 The 2-connected *double-banana* circuit can be obtained as combinatorial resultant from two K_4 graphs (left, 2-sum), and from two wheels on 4 vertices (right). Dashed lines indicate the eliminated edges, and in each case one of the two circuits is highlighted to distinguish K_4 from W_4.

Resultant tree. The inductive construction of a circuit using combinatorial resultant operations is captured by a *tree* structure, whose size influences the complexity of the algorithm. A *resultant tree* T_C for the circuit C on n vertices is a rooted binary tree with C as its root and such that: (a) the nodes of T_C are circuits; (b) circuits on level k have at most $n - k$ vertices; (c) the two children $\{C_a, C_b\}$ of a parent circuit C_c are such that $C_c = \mathrm{CRes}(C_a, C_b, e)$, for some common edge e, and (d) the leaves are complete graphs on 4 vertices. The depth of the tree is at most $n - 4$, and its size may be anywhere between linear to exponential in n. The best case occurs when the resultant tree is path-like, with each internal node having a K_4 leaf. The worst case *could be* a complete binary tree: each internal node at level k would combine two circuits with the same number $n - k - 1$ of vertices into a circuit with $n - k$ vertices. Sporadic examples of small balanced combinatorial resultant trees exist (e.g. for K_{33}-plus-one), but it remains an open problem to find infinite families of such examples. Even if such a family would be found, it is still conceivable that alternative, non-balanced combinatorial resultant trees could yield the same circuit. Answers to the following questions would refine the algorithm's analysis.

▶ **Open Problem 2.** *Are there infinite families of circuits with only balanced combinatorial resultant trees?*

▶ **Open Problem 3.** *Characterize the circuits produced by the worst-case size of the combinatorial resultant tree.*

4 Preliminaries: the Cayley-Menger ideal and its circuit polynomials

In preparation for the proof of Theorem 2, we turn now to the algebraic aspects of our problem and review concepts from polynomial ideals and algebraic matroids. We define *circuit polynomials* in the 2D Cayley-Menger ideal and make the connection with combinatorial rigidity circuits. Complete details and proofs appear in [21].

Notations and conventions. To keep the extended abstract focused, we avoid giving the most general definitions unless necessary. We restrict the presentation to the field of rational numbers $\mathbb{Q}$, to the set of variables $X_n = \{x_i : 1 \leq i \leq n\}$ (or $X_n = \{x_{i,j} : 1 \leq i < j \leq n\}$ in the Cayley-Menger setting) and to polynomial rings $R = \mathbb{Q}[X]$ over sets of variables $X \subset X_n$. The *support* $\operatorname{supp} f$ of a polynomial $f \in \mathbb{Q}[X_n]$ is the set of indeterminates appearing in f.

Polynomial ideals. A set of polynomials $I \subset \mathbb{Q}[X]$ is an *ideal of* $\mathbb{Q}[X]$ if it is closed under addition and multiplication by elements of $\mathbb{Q}[X]$. If $I \subset \mathbb{Q}[X]$ is an ideal and $X' \subset X$ is a subset of variables, then $I \cap \mathbb{Q}[X']$ is also an ideal, called the *elimination ideal* of I with eliminated variables $X \setminus X'$. A *generating set* for an ideal is a set $S \subset \mathbb{Q}[X]$ such that every polynomial in the ideal is an algebraic combination (addition and multiplication) of elements in S with coefficients in $\mathbb{Q}[X]$. *Hilbert Basis Theorem* (see [7]) guarantees that every ideal in a polynomial ring has a finite generating set. Ideals generated by a single polynomial are called *principal*. An ideal I is a *prime* ideal if, whenever $fg \in I$, then either $f \in I$ or $g \in I$. A polynomial is *irreducible* (over $\mathbb{Q}[X]$) if it cannot decomposed into a product of non-constant polynomials in $\mathbb{Q}[X]$. A principal ideal of $\mathbb{Q}[X]$ is prime iff it is generated by an irreducible polynomial. However, an ideal generated by two or more irreducible polynomials is not necessarily prime. We'll make use of the following well-known result.

▶ **Proposition 7.** *If I is a prime ideal of $\mathbb{Q}[X]$ and $X' \subset X$ is non-empty, then the elimination ideal $I \cap \mathbb{Q}[X']$ is prime.*

The theory of Gröbner bases [5, 7] gives a general framework for computing generating sets of an ideal. Gröbner bases also give a general approach for computing elimination ideals: if $\mathcal{G}$ is a Gröbner basis for I with respect to an *elimination order* (see Exercises 5 and 6 in §1 of Chapter 3 in [7]), e.g. the lexicographic order $x_{i_1} > x_{i_2} > \cdots > x_{i_n}$, then the elimination ideal $I \cap \mathbb{Q}[x_{i_{k+1}}, \ldots, x_{i_n}]$ which eliminates the first k indeterminates from I in the specified order has $\mathcal{G} \cap \mathbb{Q}[x_{i_{k+1}}, \ldots, x_{i_n}]$ as its Gröbner basis.

Resultants. The resultant can be introduced in several equivalent ways [11]. Here we use its definition as the determinant of the Sylvester matrix. Given two polynomials f and g in one variable x, of degrees r, resp. s, the Sylvester matrix is an $(r+s) \times (r+s)$ matrix whose entries are a special arrangement of the coefficients of f and g; the precise formulation is given in the full paper [21]. If the coefficients of f, g are themselves polynomials in a ring $R = \mathbb{Q}[X]$, i.e. $f, g \in R[x]$, then the resultant $\operatorname{Res}(f, g, x) \in \mathbb{Q}[X]$ is a polynomial in the coefficients' variables but not in x. In short, the resultant *eliminates* the variable x. A proof of the following proposition can be found in [7, pp. 167].

▶ **Proposition 8.** *Let I be an ideal of $R[x]$ and $f, g \in I$. Then $\operatorname{Res}(f, g, x)$ is in the elimination ideal $I \cap R$.*

Algebraic matroid of a prime ideal. Intuitively, a collection of variables is *independent* if it is not constrained by any polynomial in the ideal, and *dependent* otherwise. Formally, let I be a *prime ideal* of the polynomial ring $\mathbb{Q}[X_n]$. We define the *algebraic matroid* $\mathcal{A}(I)$ of I on the ground set X_n by its independent sets: subsets of variables that are *not* supported by any polynomial in the ideal. Its *dependent sets* are supports of polynomials in the ideal (see [25] for some small examples).

Circuits and circuit polynomials. A *circuit* is a minimal set of variables supported by a polynomial in I, called a *circuit polynomial*. The following theorem, encompassing a result of Lovasz and Dress [8], implies that *circuit polynomials generate elimination ideals supported on circuits.*

▶ **Theorem 9.** *Let I be a prime ideal in $\mathbb{Q}[X]$ and $C \subset X$ a circuit of the algebraic matroid $\mathcal{A}(I)$. The ideal $I \cap \mathbb{Q}[C]$ is principal and generated by an irreducible circuit polynomial p_C, which is unique up to multiplication by a constant.*

The Cayley-Menger ideal and its algebraic matroid. We turn now to the Cayley-Menger setting specific to our paper, where we use variables $X_n = \{x_{i,j} : 1 \leq i < j \leq n\}$ for unknown squared distances between pairs of points. The *distance matrix* of n labeled points is the matrix of squared distances between pairs of points. The *Cayley matrix* is the distance matrix bordered by a new row and column of 1's, with zeros on the diagonal:

$$\begin{pmatrix} 0 & 1 & 1 & 1 & \cdots & 1 \\ 1 & 0 & x_{1,2} & x_{1,3} & \cdots & x_{1,n} \\ 1 & x_{1,2} & 0 & x_{2,3} & \cdots & x_{2,n} \\ 1 & x_{1,3} & x_{2,3} & 0 & \cdots & x_{3,n} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{1,n} & x_{2,n} & x_{3,n} & \cdots & 0 \end{pmatrix}$$

Cayley's Theorem says that, if the distances come from a point set in the Euclidean space $\mathbb{R}^d$, then the rank of this matrix must be at most $d+2$. Thus all the $(d+3) \times (d+3)$ minors of the Cayley-Menger matrix should be zero. These minors induce polynomials in $\mathbb{Q}[X_n]$ which generate the (n,d)-Cayley-Menger ideal. They are called the *standard generators*, are *homogeneous polynomials* with integer coefficients and are *irreducible* over $\mathbb{Q}$. The (n,d)-Cayley-Menger ideal is a *prime ideal* of dimension $dn - \binom{d+1}{2}$ [3, 12, 13, 16] and codimension $\binom{n}{2} - dn + \binom{d+1}{2}$. We work with the *2D Cayley-Menger ideal* CM_n, generated by the 5×5 minors of the Cayley matrix. Its algebraic matroid is denoted by $\mathcal{A}(\mathrm{CM}_n)$.

The following result (see [21] for a complete proof), allows us to identify the ground set of the Cayley-Menger algebraic matroid X_n with the edges of the complete graph K_n, and the circuits in the algebraic matroid (supports of circuit polynomials in CM_n), as sets of variables, with graph-theoretical rigidity circuits on n vertices.

▶ **Theorem 10.** *The Cayley-Menger algebraic matroid is isomorphic to the rigidity matroid.*

From now on, we will use the isomorphism to *move freely between subsets of variables $X \subset X_n$ and their corresponding edge sets*, and between algebraic circuits and rigidity circuits. Given a (rigidity) circuit C, we denote by p_C its corresponding *circuit polynomial.*

Resultants in the Cayley-Menger ideal. Let f, g be two polynomials in the Cayley-Menger ideal with x_{ij} one of their common variables. We treat them as polynomials in x_{ij}, therefore the coefficients are themselves polynomials in the remaining variables. The fact that the entries in the Sylvester matrix are polynomials supported exactly on the variables corresponding to the *combinatorial resultant* of the supports of f and g on elimination variable (edge) ij is what motivated the definition given in Section 3.

The following lemma, whose proof follows immediately from Proposition 8, provides the connection with combinatorial resultants.

▶ **Lemma 11.** *Let I be an ideal in $\mathbb{Q}[X_n]$, $f, g \in I$ with supports $S_f = \operatorname{supp} f$ and $S_g = \operatorname{supp} g$, and let x_{ij} be a common variable in $S_f \cap S_g$. Let $S = \operatorname{CRes}(S_f, S_g, ij) \subset X_n$ be the combinatorial resultant of the supports. Then $\operatorname{Res}(f, g, x_{ij}) \in I \cap \mathbb{Q}[S]$.*

Homogeneous properties. The standard generators of the Cayley-Menger ideal, in particular those that correspond to K_4 graphs, are obviously homogeneous. The following proposition (to be used in Algorithm 2) shows that the polynomials obtained via resultants are also homogeneous, and allows us to infer their homogeneous degrees.

▶ **Proposition 12.** *The resultant $\operatorname{Res}(f, g, x)$ of homogeneous polynomials f and g of homogeneous degree m and n is a homogeneous polynomial of degree:*

$$m \deg_x g + n \deg_x f - \deg_x f \cdot \deg_x g.$$

5 Computing resultants of circuit polynomials

We are now ready to prove our main result, Theorem 2. Algorithm 2 below describes how to obtain the circuit polynomial p_C from the circuits p_A and p_B, when $C = \operatorname{CRes}(A, B, e)$.

Algorithm 2 Circuit polynomial: resultant step.

Input: Circuits A, B and edge e such that $C = \operatorname{CRes}(A, B, e)$. Circuit polynomials p_A and p_B and elimination variable x_e.
Output: Circuit polynomial p_C for C.

```
Compute the resultant p = Res(p_A,p_B,x_e).
if p is irreducible then
    p_C = p return p_C
else
    factors = factorize p over ℚ
    factors = discard factors with support not equal to C
    if exactly one remaining factor (possibly with multiplicity) then
        p_C = the unique factor supported on C
    else
        apply a test of membership in the CM ideal on the remaining factors
        p_C = unique factor for which ideal membership test succeeded
        return p_C
    end if
end if
```

The correctness of Steps 1–4 follows from the following Lemma.

▶ **Lemma 13.** *Let C be a rigidity circuit $C = \operatorname{CRes}(A, B, e)$ obtained as a combinatorial resultant of two other circuits A and B with p_A and p_B as their circuit polynomials. Then:*

1. *The resultant $\operatorname{Res}(p_A, p_B, x_e)$ is supported on C and contained in the elimination ideal $\langle p_C \rangle$ generated by the circuit polynomial p_C.*
2. *The circuit polynomial p_C of C is an irreducible factor over $\mathbb{Q}$ of $\operatorname{Res}(p_A, p_B, x_e)$.*
3. *When $\operatorname{Res}(p_A, p_B, x_e)$ is irreducible then it will be equal to p_C.*

Proof. The resultant $\operatorname{Res}(p_A, p_B, x_e)$ is a non-constant polynomial supported on C. Since $\langle p_A, p_B \rangle \subset \operatorname{CM}_n$, by Lemma 11 we have that $\operatorname{Res}(p_A, p_B, x_e)$ is contained in the elimination ideal $\operatorname{CM}_n \cap \mathbb{Q}[C] = \langle p_C \rangle$. ◀

Steps 6–9 would not be necessary if the resultant would always be irreducible. But in general p_C will only be one of the irreducible factors over $\mathbb{Q}$ of $\mathrm{Res}(p_A, p_B, x_e)$.

▶ **Lemma 14.** *The resultant of two circuit polynomials is not always a circuit polynomial.*

Proof. We prove the Lemma with an example, which can be easily generalized. Recall from Corollary 6 that in general a rigidity circuit C can be represented as the combinatorial resultant of two circuits in more than one way. If $C = \mathrm{CRes}(C_1, C_2, e) = \mathrm{CRes}(C_3, C_4, f)$ and p_{C_i} for $i \in \{1, 2, 3, 4\}$ are the corresponding circuit polynomials, then $\mathrm{Res}(p_{C_1}, p_{C_2}, x_e)$ and $\mathrm{Res}(p_{C_3}, p_{C_4}, x_f)$ will in general be distinct elements of $\langle p_C \rangle$. The 2-connected circuit in Fig. 8 has two distinct combinatorial resultant trees. Using Prop. 12 to compute the homogeneous degrees of the resultants, we obtain degrees 8 and 48. Both resultants have the same circuit as its supporting set, hence they are both in the elimination ideal of the circuit, but only the one of homogeneous degree 8 is the circuit polynomial. ◀

In general, if two resultant steps produce two polynomials on the same rigidity circuit support, the one of higher degree must have a non-trivial factor, while the other one will possibly be irreducible, but this is not guaranteed.

▶ **Corollary 15.** *Under the assumptions of Theorem 13, the resultant* $\mathrm{Res}(p_A, p_B, x_e)$ *is a circuit polynomial if and only if it is irreducible (over* $\mathbb{Q}$*).*

This leads to the following natural question.

▶ **Open Problem 4.** *Let C, A and B be rigidity circuits such that $C = \mathrm{CRes}(A, B, e)$ with p_C, p_A and p_B the corresponding circuit polynomials. Identify sufficient conditions under which* $\mathrm{Res}(p_A, p_B, x_e)$ *is p_C.*

Since in general p_C will only be one of the irreducible factors over $\mathbb{Q}$ of $\mathrm{Res}(p_A, p_B, x_e)$, we must proceed to Step 7 in the Algorithm and factorize the resultant p. *The need for the factorization step was already encountered in the examples we have so far computed.*

Determining the circuit polynomial from the resultant. If $\mathrm{Res}(p_A, p_B, x_e)$ is not irreducible, then exactly one of its irreducible factors (over $\mathbb{Q}$) is in CM_n, and that "good" factor is precisely the circuit polynomial p_C. A "bad" factor can be discarded in two steps: by analysing its support (lines 8–11) and by an ideal membership test (lines 13-14).

Analysing the supports of the irreducible factors. The elimination ideal $\langle p_C \rangle$ is an ideal of $\mathbb{Q}[C]$, and since $\mathrm{Res}(p_A, p_B, x_e) \in \langle p_C \rangle$, any irreducible factor (over $\mathbb{Q}$) of this resultant is supported on a subset of C that is not necessarily proper. We know that at least one of these factors must be supported on exactly C, and if there is only one such factor (possibly with multiplicity), then it must be p_C. Factors with a proper support are necessarily not in the ideal (they are *independent* in the Cayley-Menger matroid) and can be discarded. Hence, if there are no other factors supported on C, the algorithm stops on line 11 and returns P_C without any additional calculations. *So far, in all the concrete examples on which we could complete the calculations with our algorithm, the "bad" factors were supported on strict subsets of C*: whether this is always true remains an intriguing open question/conjecture. Should it be true, then there will be no need for Steps 9–14 in our algorithm.

▶ **Open Problem 5.** *Identify sufficient conditions for the resultant* $\mathrm{Res}(p_A, p_B, x_e)$ *of two circuit polynomials to have exactly one factor (up to multiplicity) supported on* $\mathrm{CRes}(A, B, e)$.

Lacking a definitive answer at this time, we describe a way for the algorithm to proceed.

Ideal membership test. Let's assume that we have an irreducible factor of $\mathrm{Res}(p_A, p_B, x_e)$ that is supported on C. We will have to test it for membership in CM_n. The *ideal membership test* [7, pp. 97] invokes a Gröbner basis calculation, but it can be with any monomial order, not necessarily an elimination order, hence it has a better chance at succeeding. Further details and possible algorithm optimization ideas (which we did not have yet to employ in the experimental part reported in the next section) are described in the full paper [21].

6 Concluding remarks

In this paper we introduced the combinatorial resultant operation, analogous to the classical resultant of polynomials, and used it to derive a new algorithm for computing circuit polynomials in the 2D Cayley-Menger ideal. Our approach highlights an inherent combinatorial structure in this ideal and leads to further theoretical and algorithmic questions. To demonstrate the effectiveness of our method we conclude by listing in Table 1 the circuit polynomials that we could compute within a reasonable amount of time. The most challenging was the $K_{3,3}$-plus-one circuit, which required an extension of the method presented here: this *extended resultant* is the topic of an upcoming paper [20].

Table 1 Results: all circuit polynomials on $n \leq 6$ vertices and two circuit polynomials on $n = 7$ vertices. For the definition of Extended Resultant, see [20].

n	Circuit	Method	Comp. time (seconds)	No. terms	Hom. degree
4	K_4	Determinant	0.0008	22	3
5	Wheel on 4 vertices	Gröbner Resultant	0.02 0.013	843	8
6	2D double banana	Gröbner Resultant	0.164 0.029	1 752	8
6	Wheel on 5 vertices	Gröbner Resultant	10 857 7.07	273 123	20
6	Desargues-plus-one	Gröbner Resultant	454 753 14.62	658 175	20
6	$K_{3,3}$-plus-one	Extended Resultant	1 402	1 018 050	18
7	2D double banana $\oplus_{16}$ K_4^{1567}	Resultant	38.14	1 053 933	20
7	2D double banana $\oplus_{56}$ K_4^{4567}	Resultant	89.86	2 579 050	20

References

1 Evangelos Bartzos, Ioannis Z. Emiris, Jan Legerský, and Elias Tsigaridas. On the maximal number of real embeddings of minimally rigid graphs in R^2 , R^3 and S^2. *Journal of Symbolic Computation*, 102:189–208, 2021. doi:10.1016/j.jsc.2019.10.015.

2 Alex R Berg and Tibor Jordán. A proof of Connelly's conjecture on 3-connected circuits of the rigidity matroid. *Journal of Combinatorial Theory, Series B*, 88(1):77–97, 2003. doi:10.1016/S0095-8956(02)00037-0.

3 Ciprian S. Borcea. Point configurations and Cayley-Menger varieties, 2002. arXiv:math/0207110.

4 Ciprian S. Borcea and Ileana Streinu. The number of embeddings of minimally rigid graphs. *Discrete and Computational Geometry*, 31:287–303, February 2004. doi:10.1007/s00454-003-2902-0.

5 B. Buchberger. Ein algorithmisches Kriterium für die Lösbarkeit eines algebraischen Gleichungssystems. *Aequationes Math.*, 4:374–383, 1970. doi:10.1007/BF01844169.

6 Jose Capco, Matteo Gallet, Georg Grasegger, Christoph Koutschan, Niels Lubbes, and Josef Schicho. Computing the number of realizations of a Laman graph. *Electronic notes in Discrete Mathematics*, 61:207–213, 2017. doi:10.1016/j.endm.2017.06.040.

7 David A. Cox, John Little, and Donal O'Shea. *Ideals, Varieties, and Algorithms. An Introduction to Computational Algebraic Geometry and Commutative Algebra*. Undergraduate Texts in Mathematics. Springer, Cham, fourth edition, 2015.

8 A. Dress and L. Lovász. On some combinatorial properties of algebraic matroids. *Combinatorica*, 7(1):39–48, 1987. doi:10.1007/BF02579199.

9 I. Emiris and B. Mourrain. Computer algebra methods for studying and computing molecular conformations. *Algorithmica*, 25:372–402, 1999. doi:10.1007/PL00008283.

10 Ioannis Z. Emiris, Elias P. Tsigaridas, and Antonios Varvitsiotis. Mixed Volume and Distance Geometry Techniques for Counting Euclidean Embeddings of Rigid Graphs. In Mucherino, Antonio and Lavor, Carlile and Liberti, Leo and Maculan, Nelson, editor, *Distance Geometry. Theory, Methods, and Applications*, chapter 2, pages 23–46. Springer, New York, Heidelberg, Dordrecht, London, 2013. doi:10.1007/978-1-4614-5128-0.

11 I.M. Gelfand, M. Kapranov, and A. Zelevinsky. *Discriminants, Resultants, and Multidimensional Determinants*. Modern Birkhäuser Classics. Birkhäuser Boston, 2009. doi:10.1016/0001-8708(90)90047-Q.

12 G.Z. Giambelli. Sulle varietá rappresentate coll'annullare determinanti minori contenuti in un determinante simmetrico od emisimmetrico generico di forme. *Atti della R. Acc. Sci, di Torino*, 44:102–125, 1905/06.

13 J. Harris and L.W. Tu. On symmetric and skew-symmetric determinantal varieties. *Topology*, 23:71–84, 1984. doi:10.1016/0040-9383(84)90026-0.

14 Lebrecht Henneberg. *Die graphische Statik der starren Systeme*. B. G. Teubner, 1911.

15 John E. Hopcroft and Robert Endre Tarjan. Dividing a graph into triconnected components. *SIAM J. Comput.*, 2(3):135–158, 1973. doi:10.1137/0202012.

16 T. Józefiak, A. Lascoux, and P. Pragacz. Classes of determinantal varieties associated with symmetric and skew-symmetric matrices. *Math. USSR Izvestija*, 18:575–586, 1982. doi:10.1070/im1982v018n03abeh001400.

17 Gerard Laman. On graphs and rigidity of plane skeletal structures. *Journal of Engineering Mathematics*, 4:331–340, 1970. doi:10.1007/BF01534980.

18 Audrey Lee-St. John and Ileana Streinu. Pebble game algorithms and sparse graphs. *Discrete Mathematics*, 308(8):1425–1437, April 2008. doi:10.1016/j.disc.2007.07.104.

19 Goran Malić and Ileana Streinu. CayleyMenger - Circuit Polynomials in the Cayley Menger ideal, a GitHub repository. https://github.com/circuitPolys/CayleyMenger, 2020.

20 Goran Malić and Ileana Streinu. Circuit polynomial for the $K_{3,3}$-plus-one circuit, 2021. In preparation.

21 Goran Malić and Ileana Streinu. Combinatorial resultants in the algebraic rigidity matroid, 2021. ArXiv/2103.08432 [Math.CO]. arXiv:2103.08432.

22 James Oxley. *Matroid theory*, volume 21 of *Oxford Graduate Texts in Mathematics*. Oxford University Press, Oxford, second edition, 2011.

23 Zvi Rosen. *Algebraic Matroids in Applications*. PhD thesis, University of California, Berkeley, 2015. URL: https://digitalassets.lib.berkeley.edu/etd/ucb/text/Rosen_berkeley_0028E_15261.pdf.

24 Zvi Rosen. algebraic-matroids, a GitHub repository. https://github.com/zvihr/algebraic-matroids, 2017.

25 Zvi Rosen, Jessica Sidman, and Louis Theran. Algebraic matroids in action. *The American Mathematical Monthly*, 127(3):199–216, February 2020. doi:10.1080/00029890.2020.1689781.

26 Meera Sitharam and Heping Gao. Characterizing graphs with convex and connected Cayley configuration spaces. *Discrete and Computational Geometry*, 43(3):594–625, 2010. doi:10.1007/s00454-009-9160-8.

27 William T. Tutte. *Connectivity in graphs*. Toronto University Press, Toronto, 1966.

28 D. Walter and M.L. Husty. On a nine-bar linkage, its possible configurations and conditions for flexibility. In Merlet J.-P. and M. Dahan, editors, *Proceedings of IFFToMM 2007, Besançon, France*, 2007. URL: http://geometrie.uibk.ac.at/cms/datastore/husty/A681.pdf.

Parameterized Complexity of Quantum Knot Invariants

Clément Maria
INRIA Sophia Antipolis-Méditerranée, France

Abstract

We give a general fixed parameter tractable algorithm to compute quantum invariants of links presented by planar diagrams, whose complexity is singly exponential in the carving-width (or the tree-width) of the diagram.

In particular, we get a $O(N^{\frac{3}{2}\,\mathrm{cw}}\mathrm{poly}(n)) \in N^{O(\sqrt{n})}$ time algorithm to compute any Reshetikhin-Turaev invariant – derived from a simple Lie algebra $\mathfrak{g}$ – of a link presented by a planar diagram with n crossings and carving-width cw, and whose components are coloured with $\mathfrak{g}$-modules of dimension at most N. For example, this includes the N^{th}-coloured Jones polynomial.

2012 ACM Subject Classification Theory of computation → Fixed parameter tractability; Theory of computation → Computational geometry; Mathematics of computing → Graphs and surfaces

Keywords and phrases computational knot theory, parameterized complexity, quantum invariants

Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.53

Related Version *Full Version*: `https://arxiv.org/abs/1910.00477`

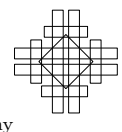
Funding *Clément Maria*: This work has been partially supported by the ANR project ANR-20-CE48-0007-01 (AlgoKnot).

1 Introduction

In geometric topology, testing the topological equivalence of knots (up to isotopy) is a fundamental yet remarkably difficult algorithmic problem.

A main approach is to compare knots by properties depending on their topological types, called *invariants*. Starting with the introduction by Jones [15] in the 1980s of a new polynomial invariant of knots, we have witnessed the birth of a new domain of low dimensional topology called *quantum topology*. From the study of quantum groups [6, 14] in algebra, topologists have designed new families of topological invariants for knots, links, and 3-manifolds, such as the Reshetikhin-Turaev invariants [25]. In practice, these *quantum invariants* have shown outstanding discriminative properties for non-equivalent knots and links, *e.g.*, in the composition of knot census databases [4] that are fundamental to the mathematical work in geometric topology. The infinite families of quantum invariants have even been conjectured to be complete, *i.e.*, that no non-equivalent knot may have all identical quantum invariants. They are also at the heart of deep mathematical conjectures in the field [7, 8, 16, 24].

Consequently, in order to effectively distinguish between knots or verify experimentally mathematical conjectures, efficient algorithms to compute quantum invariants are of strong interest. However, even the simplest quantum invariants, such as the Jones polynomial [13], are #P-hard to compute. A successful approach towards practical implementations has been the introduction of *parameterized complexity* to low dimensional topology. Independently, computing the Jones polynomial [19] and the HOMFLYPT polynomial [3] have been shown to admit fixed parameter tractable algorithms in the tree-width of the input link diagrams. Note that similar techniques have been applied to quantum invariants of 3-manifolds, such




as the Turaev-Viro-Barrett-Westbury invariants of triangulated 3-manifolds [5, 30], where other types of parameters have also been considered [22]. These algorithms led to significant speed-ups in practice, with applications in experimental mathematics [4, 21].

Contribution. We introduce an algorithm to compute quantum invariants derived from ribbon categories [25, 29], taking into account the carving-width of the input link diagram.

▶ **Theorem 1.** *Fix a strict ribbon category $\mathcal{C}$ of $\mathbb{Z}[q]$-modules, and free modules $V_1, \dots, V_m \in \mathcal{C}$ of dimension bounded by N. The problem:*

> Quantum invariant at $\mathcal{C}, V_1, \dots, V_m$:
> **Input**: *m-components link L, presented by a diagram $D(L)$,*
> **Output**: *quantum invariant $J^{\mathcal{C}}_L(V_1, \dots, V_m)$*

can be solved in $O(\text{poly}(n)N^{\frac{3}{2}\,\text{cw}}) \in N^{O(\sqrt{n})}$ machine operations, with $O(N^{\text{cw}} + n)$ memory words, where n and cw *are respectively the number of crossings and the carving-width of the diagram $D(L)$.*

In particular, this implies that, up to some easily computable re-normalisation, computing any fixed Reshetikhin-Turaev invariant derived from a simple Lie algebra $\mathfrak{g}$ is *fixed parameter tractable* (complexity class FPT) in the carving-width (and also the tree-width) of the input link diagram.

To compare with the state-of-the-art, the only known parameterized and sub-exponential time algorithms for quantum knot invariants are the Jones polynomial [19] ($\mathfrak{g} = \mathfrak{sl}(2, \mathbb{C})$ and $N = 2$) and the HOMFLYPT polynomial [3] (which interpolates the invariants for $\mathfrak{g} = \mathfrak{sl}(m, \mathbb{C})$, for all m). So,

1. Theorem 1 generalises these results (fixed parameter tractable algorithm and sub-exponential time algorithm) to *all the infinitely many* dimensions $N \geq 2$ – the so called *coloured* Jones polynomials – quantum invariants, and in general to all coloured invariants derived from other simple Lie algebras $\mathfrak{g}$.
2. It also generalises Burton's result on the parameterized complexity of the HOMFLYPT polynomial, to offer a parameterized solution to the *coloured* HOMFLYPT polynomials by interpolation; see [23].
3. More generally, for knot theorists, the algorithm of Theorem 1 is a low exponent ($\frac{3}{2}$ cw) singly exponential algorithm for quantum invariants[1]. In practice, and considering past experience in 3-manifolds, this work offers a practical algorithm in order to verify fundamental mathematical conjectures [7, 8, 16, 24] experimentally, and further tools to distinguish between non-equivalent knots in the constitution of knot censuses [4] thanks to the outstanding discriminative power of quantum invariants.
4. Finally, for computer scientists, the Jones polynomial has been studied in detail as it has a very rich complexity theory: it is $\#P$-hard [13] but admits an FPT and sub-exponential time algorithm [19] ; it is also one of the few known natural BQP-complete problems in quantum computing [1]. This article completes the computational complexity picture on quantum invariants by showing that very large families of them admit an FPT and sub-exponential algorithm.

[1] Note that previous algorithms [19] are expressed in terms of tree-width, which is proportional but not equal to the carving-width, in consequence exponents are not directly comparable ; see Theorem 4.

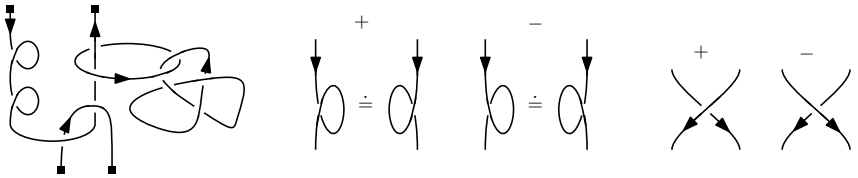


Figure 1 Left: Diagram of a 4-components oriented tangle; leftmost component has framing +2. Right: Positive/negative twists and crossings. The $\doteq$ symbol is an equivalence of diagrams.

The key approach of this work is the use of the abstract construction of quantum invariants via *ribbon categories*, in order to state the complexity result in full generality. This way, we abstract ourselves from the specificity of each singular invariant to give a general algorithm.

In Section 2 we recall the general definition of quantum invariants derived from ribbon categories, and notions of parameterized complexity. In Section 3 we give the general dynamic programming approach for the parameterized algorithm, relying on a decomposition of the knot diagram. In Section 4 we detail the topological and algebraic operations necessary to implement the dynamic programming procedure ; the connection between topology and algebra relies on graphical calculus. Finally, in Section 5, we implement the algebraic operations with matrix multiplications, and analyse their complexity and the complexity of the whole algorithm. In conclusion, we prove that all arithmetic operations have polynomial time complexity and prove the main Theorem 1 in Section 6. Finally, in [23], we implement the algorithm and verify mathematical conjectures on formally intractable quantum invariants.

2 Background

We introduce the necessary notions from knot theory and parameterized complexity.

Tangles and their diagrams. A *tangle* is a piecewise linear embedding of a collection of arcs and circles into $\mathbb{R}^2 \times [0,1]$, such that the arcs' endpoints belong to the top or bottom boundaries $\mathbb{R}^2 \times \{0\}$ and $\mathbb{R}^2 \times \{1\}$. A tangle intersecting i times $\mathbb{R}^2 \times \{0\}$ and j times $\mathbb{R}^2 \times \{1\}$ is an (i,j)-tangle.

A *link* is a tangle whose connected components are all closed curves (a $(0,0)$-tangle), and a *knot* is a 1-component link. An *orientation* on a tangle is an orientation of each tangle component. Two tangles are equivalent *iff* they differ by an ambient isotopy of $\mathbb{R}^2 \times [0,1]$ maintaining the boundary fixed.

A tangle *diagram* is a projection of the tangle into the plane, induced by a projection of $\mathbb{R}^2 \times [0,1]$ into $\mathbb{R} \times [0,1]$, sending $\mathbb{R}^2 \times \{0\}$ and $\mathbb{R}^2 \times \{1\}$ to $\mathbb{R} \times \{0\}$ and $\mathbb{R} \times \{1\}$ respectively. In a tangle diagram, the only multiple points are *crossings*, at which one section of the tangle crosses under or over another one transversally. We consider diagrams of $(0,0)$-tangles (*i.e.*, link diagrams) as drawn on the sphere S^2. Component orientations are pictured with arrow heads, and a $k \in \mathbb{Z}$ *framing* is pictured by k *positive twists* if $k > 0$, and k *negative twists* is $k < 0$. See Figure 1. We refer to [17] for more details on knot theory.

Graphical calculus on coloured tangles. We work in the category $\mathcal{C}$ of free finitely generated $\mathcal{R}$-modules, for a commutative ring $\mathcal{R}$, with their usual tensor product $\otimes$ and dual V^* for every object V. Additionally, some fixed special objects and morphisms are distinguished and play an important role:

(a) a finite set of objects $\{V_i : i \in I\} \subset \mathcal{C}$, called *colours*,
(b) for every pair of colours V_i, V_j, an isomorphism $c_{V_i,V_j} : V_i \otimes V_j \to V_j \otimes V_i$ and its inverse $c^{-1}_{V_i,V_j}$, called *braidings*,
(c) for every colour V_i, two morphisms $b_{V_i} : \mathbb{1} \to V_i \otimes V_i^*$ and $d_{V_i} : V_i^* \otimes V_i \to \mathbb{1}$, called respectively *co-evaluation* and *evaluation*, and
(d) for every colour V_i, an isomorphism $\theta_{V_i} : V_i \to V_i$ and its inverse $\theta^{-1}_{V_i}$, called a *twist*.

We call the ring $\mathcal{R}$, seen as a module over itself, the *unit module*, and denote it by $\mathbb{1}$. Note that the set of morphisms $\mathrm{Hom}_{\mathcal{C}}(\mathbb{1}, \mathbb{1})$ has the structure of a commutative ring, isomorphic to $\mathcal{R}$, *i.e.*, every module morphism $\mathbb{1} \to \mathbb{1}$ is a multiplication by a scalar $\tau \in \mathcal{R}$. By convention, the tensor product of zero objects is equal to $\mathcal{R}$.

Such category of modules, with *braidings*, *twists*, *evaluations* and *co-evaluations*, is a *strict ribbon category* if the objects and morphisms satisfy a certain set of additional equations. The theory of strict ribbon category is outside the scope of this article, and we only use the fact that strict ribbon categories define topological invariants of links (Theorem 2 below). We refer the interested reader to [29] for the general theory.

Fix a strict ribbon category $\mathcal{C}$. A *colouring* of a link L, with m ordered components $L_1, \ldots, L_m$, is an assignment of a colour $V_i \in \mathcal{C}$, $1 \leq i \leq m$, to every component L_i of L.

A link diagram is in *standard position* if it can be decomposed into the following pieces: vertical strands, positive and negative crossings, positive and negative right twists, and caps and cups. See Figure 3 for a Hopf link in standard position. Any link diagram can be isotoped into standard position without modifying the diagram combinatorially [29].

Figure 2 presents the conversion from oriented coloured tangles to $\mathcal{C}$-morphisms, called *graphical calculus*, and Figure 3 gives a full example on the Hopf link. Specifically, given a coloured link diagram $D(L)$ in standard position, the graphical calculus turns the diagram into a morphism, following a set of conversion rules (from link diagram to morphism) and equivalence relations, pictured graphically in Figure 2, and described below:

Figure 2a (i)&(ii) A morphism $f : U \to V$ in $\mathcal{C}$ is represented graphically by a box, aligned with x- and $y-$axis, called a *coupon*, with an incoming vertical V-coloured strand (top) and an outgoing vertical U-coloured strand (bottom). The identity morphism id_V is equivalent to a vertical downward-oriented V-coloured strand. Note that morphisms, and more generally entire diagrams, are read *bottom-to-top*.

Figure 2a (iii) reversing a component orientation changes a colour V to its dual V^*. See also Figure 2b for equivalent representations of a morphism changing strands orientations.

Figure 2c (i)&(ii) two parallel strands coloured U (left) and V (right) respectively are equivalent to a single strand coloured $U \otimes V$. Two side-by-side coupons for morphisms h_1 (left) and h_2 (right) are equivalent to a single coupon for morphism $h_1 \otimes h_2$.

Figure 2c (iii) a coupon for morphism g on top of a coupon for morphism f is equivalent to their composition $g \circ f$,

Figure 2d (i)&(ii) a positive crossing is equivalent to a braiding morphism, a negative crossing is equivalent to the inverse of the braiding morphism. See also Figure 2d (iii) for a realisation of $c^{-1} \circ c = \mathrm{id}$ as a Reidemeister move.

Figure 2e (i)&(ii) positive and negative twists are equivalent to the twist morphism and its inverse respectively,

Figure 2f (i)&(ii) caps and cups are equivalent to evaluation and co-evaluation morphisms respectively.

Figure 2g graphical and algebraic definition of the dual morphism $f^*\colon V^* \to U^*$ of a morphism $f\colon U \to V$.

Figure 2h Sliding coupons along strands does not change the corresponding morphism.

The morphisms are applied to the objects colouring the entering and leaving strands. Note that, by convention, morphisms are composed bottom-up. Additionally, we can reverse the orientation of strands by dualising their colours if convenient. Figure 3 gives the morphism associated to the Hopf link coloured with objects U and V.

Consequently, for a category $\mathcal{C}$, graphical calculus associates to any coloured link, seen as a $(0,0)$-tangle, a morphism $\mathbb{1} \to \mathbb{1}$. More generally, it associates to a coloured (i,j)-tangle a morphism $U_1 \otimes \ldots \otimes U_i \to V_1 \otimes \ldots \otimes V_j$, for the $\mathcal{C}$-objects U_k and V_ℓ colouring the bottom and top endpoints of the tangle respectively.

If the ordered components of a link L are coloured $V_1, \ldots, V_m$, this morphism is written:

$$J_L^{\mathcal{C}}(V_1, \ldots, V_m) \in \mathrm{Hom}_{\mathcal{C}}(\mathbb{1}, \mathbb{1}).$$

Strict ribbon categories produce topological invariants, called *quantum invariants*:

▶ **Theorem 2** ([25, 29]). *Let L be an m-components link, and D_1 and D_2 two arbitrary diagrams of L in S^2. Let $V_1, \ldots, V_m \in \mathcal{C}$ be a colouring of the components of L, and $J_{D_1}^{\mathcal{C}}(V_1, \ldots, V_m)$ and $J_{D_2}^{\mathcal{C}}(V_1, \ldots, V_m)$ the two quantities obtained by graphical calculus on D_1 and D_2 respectively. Then:*

$$J_{D_1}^{\mathcal{C}}(V_1, \ldots, V_m) = J_{D_2}^{\mathcal{C}}(V_1, \ldots, V_m) = J_L^{\mathcal{C}}(V_1, \ldots, V_m).$$

Using $\mathrm{Hom}(\mathbb{1}, \mathbb{1}) \cong \mathcal{R}$, we identify the invariant $J_L^{\mathcal{C}}(V_1, \ldots, V_k)$ to a scalar in $\mathcal{R}$.

Graph parameters. The *carving-width*, also known as *congestion*, is a graph parameter introduced by Seymour and Thomas [28].

▶ **Definition 3.** *Let $\mathcal{G} = (X, E)$ be a graph on n vertices, with loops and multiple edges. Let $\mathcal{T}$ be an unrooted binary tree, with all internal nodes of degree 3, and with n leaves.*

An embedding *ϕ of $\mathcal{G}$ into $\mathcal{T}$ is a bijective mapping between the nodes X of $\mathcal{G}$ and the leaves of $\mathcal{T}$. Every edge e of $\mathcal{T}$ induces a partition of the vertices of $\mathcal{G}$ into two sets, $X = X_e^{(1)} \sqcup X_e^{(2)}$, inherited from the partition of $\mathcal{T} \smallsetminus e$ into two trees. Let $\mathrm{w}(e)$ denote the number of edges in $\mathcal{G}$ between $X_e^{(1)}$ and $X_e^{(2)}$, called the weight of e. See Figure 4.*

The width *of an embedding $(\mathcal{T}, \phi)$ is the maximal weight of a tree edge:* $\max_{e \text{ edge of } \mathcal{T}} \mathrm{w}(e)$.

The carving-width $\mathrm{cw}(\mathcal{G})$ *of a graph $\mathcal{G}$ is the minimal width over all its embeddings into binary trees. The* carving-width $\mathrm{cw}(D(L))$ of a link diagram $D(L)$ *is the carving-width of the 4-valent planar graph it realises (we use the same notation).*

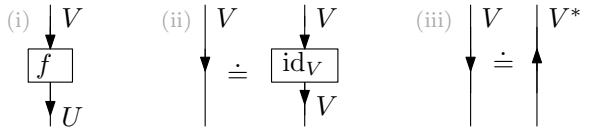
The carving-width $\mathrm{cw}(\mathcal{G})$ of a graph $\mathcal{G}$ is closely related to its *tree-width* [26] $\mathrm{tw}(\mathcal{G})$, which plays a major role in combinatorial algorithms.

▶ **Theorem 4** (Theorem 1 of [2]). *Let $\mathcal{G}$ be a graph of maximal degree δ. Then,*

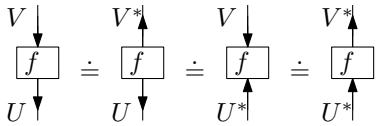
$$\frac{2}{3}(\mathrm{tw}(\mathcal{G}) + 1) \leq \mathrm{cw}(\mathcal{G}) \leq \delta(\mathrm{tw}(\mathcal{G}) + 1). \qquad \textit{For tangle diagrams } \delta \leq 4.$$

Carving-width has several advantages over tree-width, and has been successfully used in low dimensional topology [11, 12, 20, 27].

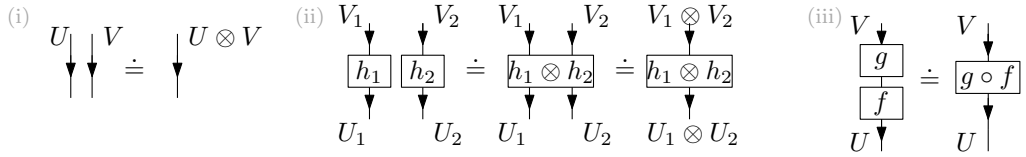
First, optimal tree embeddings of planar graphs can be realised topologically, as stated below. A tree embedding $(\mathcal{T}, \phi)$ of $\mathcal{G}$ is *bond* if the two vertex sets $X_e^{(1)}$ and $X_e^{(2)}$ from the cut associated to an edge e of $\mathcal{T}$ induce connected sub-graphs in $\mathcal{G}$.



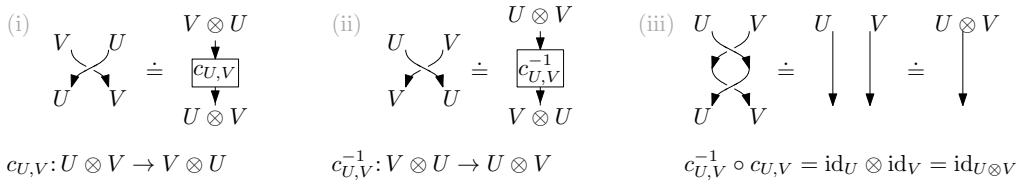
(a) (i) Graphical representation of morphism $f\colon U \to V$ via stands and coupon. (ii) Equivalence strand and identity. (iii) Dualisation and strand orientation.



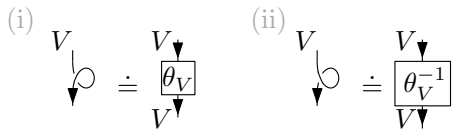
(b) Four equivalent representations of morphism $f\colon U \to V$ with all possible strands orientations.



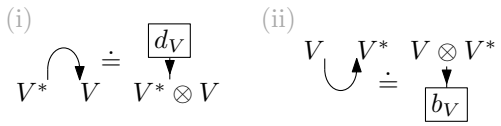
(c) (i) & (ii) Side-by-side coloured strands and morphisms is equivalent to tensor product of colours and morphisms. (iii) Morphisms on top of one another is equivalent to composition.



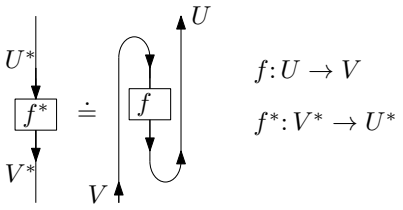
(d) (i) Positive and (ii) negative crossings are equivalent to the braiding morphism and its inverse, respectively. As illustration, the equation $c^{-1} \circ c = \mathrm{id}$ is equivalent to a Reidemeister move II.



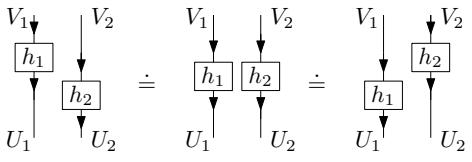
(e) (i) Positive and (ii) negative twists (framing) are equivalent to the twist morphism and its inverse, respectively.



(f) (i) Local maxima (caps) and (ii) local minima (cups) are equivalent to evaluation and coevaluation morphisms.



(g) Definition of the dual f^* of a morphism f, as $f^* := (d_V \otimes \mathrm{id}_{U^*}) \circ (\mathrm{id}_{V^*} \otimes f \otimes \mathrm{id}_{U^*}) \circ (\mathrm{id}_{V^*} \otimes b_U)$.



(h) Invariance by sliding coupons along strands, *i.e.*, $(h_1 \otimes \mathrm{id}_{U_2}) \circ (\mathrm{id}_{U_1} \otimes h_2) = h_1 \otimes h_2 = (\mathrm{id}_{U_1} \otimes h_2) \circ (h_1 \otimes \mathrm{id}_{U_2})$.

Figure 2 Formal rules and some equations for graphical calculus.

▶ **Theorem 5** ([28, Theorem 5.1]). *Let $\mathcal{G}$ be 2-connected with at least two nodes. If $\mathcal{G}$ has carving-width* cw *then there exists a bond tree embedding of $\mathcal{G}$ of width* cw.

This theorem applies to link diagrams [27]. We interpret a bond tree embedding of a planar graph (on the sphere S^2) as a collection of disjoint Jordan curves $\lambda_e \subset S^2$, one for each edge e of $\mathcal{T}$, realising the cut $X_e^{(1)} \sqcup X_e^{(2)}$.

$$[(d_U \otimes d_{V^*}) \circ (\mathrm{id}_{U^*} \otimes c_{U,V} \otimes \mathrm{id}_{V^*}) \circ (\mathrm{id}_{U^*} \otimes c_{V,U} \otimes \mathrm{id}_{V^*}) \circ (b_{U^*} \otimes b_V)] : \mathbb{1} \to \mathbb{1}$$

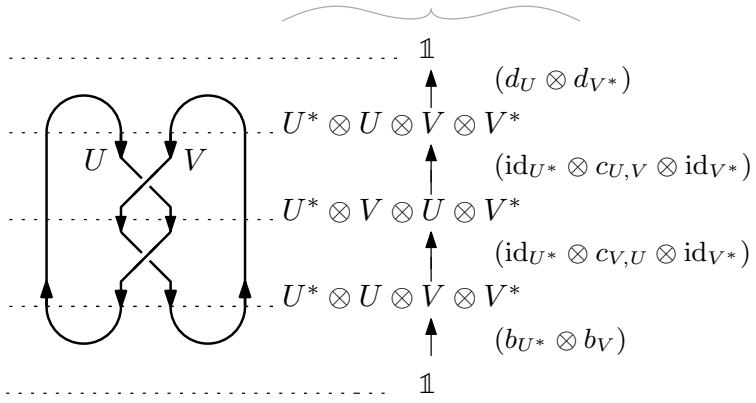


Figure 3 Application of graphical calculus to the Hopf link coloured by objects U and V from a strict ribbon category, leading to a $\mathbb{1} \to \mathbb{1}$ morphism by composition. Composition is bottom-to-top, the expression of morphisms is given on the right, and correspond to the two cups (co-evaluations), then the crossings (braidings), then the two caps (evaluations).

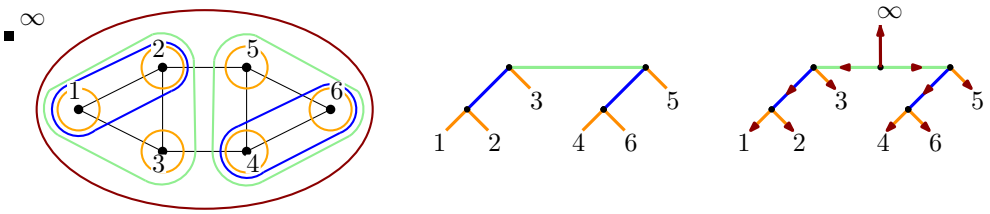


Figure 4 Left/middle: 6-vertex graph on S^2 with width 3 bond tree embedding (tree and Jordan curves). Note that the two green curves represent the same cut on S^2. Right: rooting of the tree by adding a node at ∞.

For planar graphs, a bond tree embedding of minimal width can be computed in polynomial time [10, 28]. No such result is known for tree-width. By the planar separator theorem [18], the carving-width of a planar graph with n-vertices is in $O(\sqrt{n})$.

3 Fixed parameter tractable algorithm via graphical calculus

Let $\mathcal{C}$ be a strict ribbon category, and let L be an oriented link with m components $L_1, \ldots, L_m$. Let $D(L)$ be an oriented link diagram of L, where each link component L_i is coloured by an object V_i from the category $\mathcal{C}$, such that graphical calculus gives an isotopy invariant of L associated to its colouring, as described in Theorem 2.

It follows from the definition of graphical calculus, and particularly equality Figure 2c (i)&(ii), that the quantum invariant of a separable link $L \cup L'$ is the product of the invariants of L and L', such that they can be computed separately. W.l.o.g. we assume that the diagram $D(L)$ is connected as a graph, and has at least 2 crossings, not all twists. If any of the two last requirements are not met, which can be checked in linear time, the diagram represents a possibly framed trivial knot, for which quantum invariants can be deduced directly. Consequently, all crossings of diagrams have degree four, with at most one self-edge ; see Figure 5. These properties are used in the complexity analysis Section 5.2.

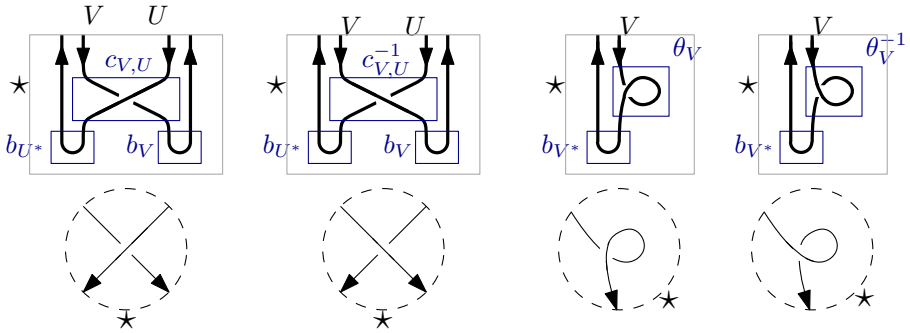


Figure 5 Morphisms associated to neighbourhoods of crossings, *i.e.*, morphisms at leaves of a tree embedding. From left to right, the corresponding equations are given by Eq. (1), (2), (3), and (4) of Section 3.3. The starred point is selected such that only these four morphisms are encountered (*i.e.*, between the two arrow heads for crossings, and on the side of the loop for twists).

3.1 Tree embedding of link diagrams

Let $(\mathcal{T}, \phi)$ be a bond tree embedding of the planar graph of $D(L)$ in S^2. We add a fake, disconnected, *node at infinity* to the graph $D(L)$, and we add a corresponding new leaf to the tree embedding $\mathcal{T}$ by subdividing an arbitrary tree edge, and connecting a leaf to it. Topologically, the Jordan curve in a neighbourhood of the node at infinity encircles the entire graph $D(L)$. We root the tree $\mathcal{T}$ to the new inner node to which the new leaf is attached ; see Figure 4. All edges of $\mathcal{T}$ have now a *parent* and *child* endpoint.

Let e be an edge of $\mathcal{T}$ with child node z, and Z the set of crossings mapped to the leaves of the subtree $\mathcal{T}_z$ rooted at z. According to Theorem 5, there exists a Jordan curve λ_e separating Z from the rest of the diagram. All tangle diagrams are on the sphere. By convention, we draw the tangle inside the Jordan curve when we represent it on the plane.

To an edge e corresponds a $(0, \mathrm{w}(e))$-tangle T, spanned by the crossings Z and contained inside the Jordan curve λ_e. We mark an arbitrary but fixed *starred point* on λ_e, and order the endpoints of the tangle T counter-clockwise starting at that starred point. We get a $(0, \mathrm{w}(e))$-tangle by isotopically sliding all tangle endpoints to the top boundary, such that the first endpoint in the *starred point ordering* is rightmost on the top boundary. See Figure 5 for examples of $(0, \mathrm{w}(e))$-tangles at the tree leaves, and Figure 6 (left) for endpoints ordered by starred point ordering.

In the process of the algorithm below, starred points are assigned on the fly. They are used exclusively to deterministically order tangles' endpoints and process the isotopy described above ; they induce a planar isotopy of the tangle and consequently affect the computation, but lead to the same output thanks to the isotopy invariance Theorem 2.

3.2 Tree traversal algorithm

Let $D(L)$ be coloured by objects of the category $\mathcal{C}$. To every edge e of weight $\mathrm{w}(e)$ in $\mathcal{T}$, graphical calculus assigns a $\mathcal{C}$-morphism $f_e \colon \mathbb{1} \to V_1 \otimes \ldots \otimes V_{\mathrm{w}(e)}$ to the associated tangle, where $V_1, \ldots, V_{\mathrm{w}(e)}$ are the colours of the strands intersecting the Jordan curve λ_e.

The morphism associated to the half-edge at the root is a $\mathbb{1} \to \mathbb{1}$ morphism, because the corresponding Jordan curve does not intersect the link diagram. This morphism gives the invariant $J^{\mathcal{C}}_L \in \mathcal{R}$ of Theorem 2. All edge morphisms are computed recursively following a depth first traversal of $\mathcal{T}$. We describe the base morphisms assigned to the edges whose child node is a leaf, and we describe an algorithm for inner edges in the next section.

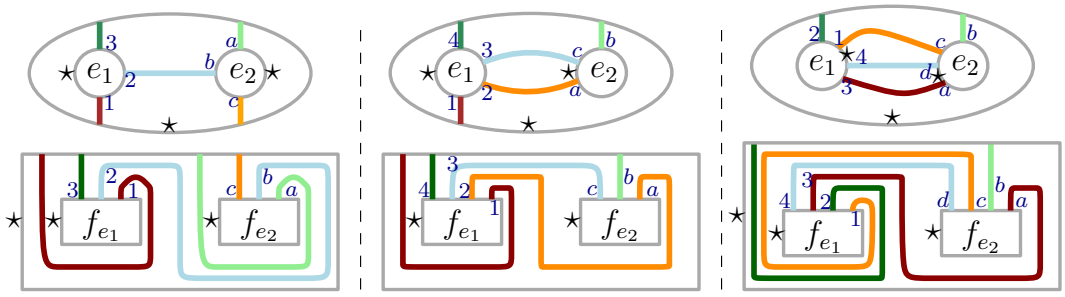


Figure 6 Merging two sub-trees. Top: Planar embeddings of the diagram with Jordan curves λ_{e_1}, λ_{e_2} (inner circles) and λ_e (outer circle), depending on the relative position of the starred points for λ_{e_1} and λ_{e_2}. The bold lines connecting the Jordan curves represent multiple parallel strands connecting the corresponding tangles. Bottom: Coupons for f_{e_1}, f_{e_2} and f_e (outer coupon) obtained after plane isotopy. The starred point for λ_e is selected so as to restrict to these three cases.

3.3 Morphisms at the leaves

The tree leaves are in bijection with tangles made by small neighbourhoods around single crossings. All crossings being of degree 4, and up to reorientation of the strands which algebraically consists of dualising colours, we get four base morphisms: one positive or negative crossing, or one positive or negative twist (with a self-edge),

$$(\mathrm{id}_{U^*} \otimes c_{V,U} \otimes \mathrm{id}_{V^*}) \circ (b_{U^*} \otimes b_V) \quad (1) \qquad (\mathrm{id}_{U^*} \otimes c_{V,U}^{-1} \otimes \mathrm{id}_{V^*}) \circ (b_{U^*} \otimes b_V) \quad (2)$$

$$(\mathrm{id}_{V^*} \otimes \theta_V) \circ b_{V^*} \quad (3) \qquad (\mathrm{id}_{V^*} \otimes \theta_V^{-1}) \circ b_{V^*} \quad (4)$$

They correspond graphically to the diagrams in Figure 5 (nodes of degree 4 are crossings, nodes of degree 2 are twists).

3.4 Merging morphisms at tree nodes

Every inner node x of $\mathcal{T}$ is the parent node of two edges e_1 and e_2, and the child of an edge e. Given the morphisms f_{e_1} and f_{e_2} for edges e_1 and e_2 respectively, we construct the morphism f_e for edge e. This is the fundamental operation of the dynamic programming algorithm ; the end of Section 3 and Section 4 are dedicated to the design of an algorithm for this operation.

First, note that the starred point ordering of the strands intersecting λ_{e_1} and λ_{e_2} leads to three configurations when representing morphisms f_{e_1} and f_{e_2} with coupons ; see Figure 6 where thick lines represent sets of parallel tangle strands. By hypothesis, morphisms on tree edges have domain $\mathbb{1}$. The coupons for f_{e_1}, f_{e_2}, and f_e (the outer coupon) are obtained by a plane isotopy forcing the strands to intersect coupons on their top side, and putting starred points on the coupons' left sides. More specifically, the isotopies are implemented by rotating annulus neighbourhoods of the Jordan curves (grey circles in Figure 6) in order to position the starred point on the left, then straightening the strands.

The starred point of the outer coupon f_e is selected so as to restrict to the three configurations of Figure 6.

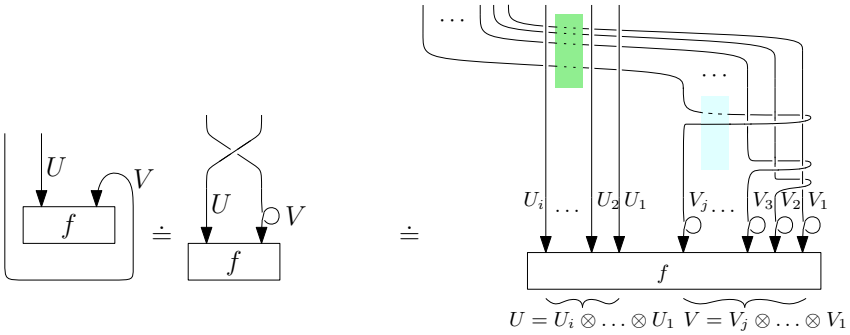


Figure 7 Sliding of a $V = V_j \otimes \ldots \otimes V_1$-coloured strand under an f-coupon by underlying knot isotopy. The operation composes f with a consecutive sequence of j twists θ_{V_ℓ}, of $j(j-1)$ crossings $c^{\pm}_{V_i,V_j}$, and ij crossings $c^{\pm}_{U_k,V_\ell}$.

4 Factorisation of morphisms at tree nodes

Given the morphisms f_{e_1} and f_{e_2} in Figure 6, we describe graphically a factorisation scheme to obtain the morphism f_e.

4.1 Sliding and canonical form

The *canonical form* for morphisms to be merged is depicted in the top left corner of Figure 8. It consists of two side-by-side morphisms g_1 and g_2, bridged by parallel strands coloured $U_1, \ldots U_k$. All other strands go up vertically.

Given morphisms f_{e_1} and f_{e_2} in Figure 6, we obtain a canonical form by sliding strands, wrapping clockwise around the coupons, under the coupons. For example, in the top right case of Figure 6, we slide strand 1 under the f_{e_1}-coupon, and strands a and b under the f_{e_2}-coupons.

The details of the operation are depicted in Figure 7, where the V-strand wraps clockwise around the f-coupon, and f is a $\mathbb{1} \to U \otimes V$ morphism. Sliding the V-strand under the coupon by tangle isotopy produces a positive twist θ_V and a positive crossing $c_{V,U}$.

More specifically, decomposing further in Figure 7, let $U = U_i \otimes \ldots \otimes U_1$ be the tensor product of the colours of i parallel strands, and $V = V_j \otimes \ldots \otimes V_1$ the tensor product of j parallel strands wrapping clockwise around the f coupon. As depicted in the figure, sliding the j strands under f induces:

- a twist θ_{V_ℓ} on each of the V_ℓ-coloured strands, $1 \leq \ell \leq j$,
- a sequence of $j(j-1)$ positive and negative crossings of type $c^{\pm}_{V_\ell,V_k}$, followed by
- a sequence of ij positive crossings of type c_{V_ℓ,U_k}.

We obtain the morphisms g_1, g_2 of the canonical form (Figure 8) by factorising the morphisms f_{e_1} and f_{e_2} with these sequences of twists and crossings, after the sliding operation.

4.2 Factorisation of the canonical form

Figure 8 pictures two factorisation schemes for side-by-side morphisms g_1 and g_2 in canonical form, bridged by k parallel strands coloured $U_1, \ldots, U_k$. Denote by cw the carving-width of the link diagram, and assume the tree embedding $(\mathcal{T}, \phi)$ has width cw.

We distinguish between two cases, depending on the value of k, which is necessary to obtain the factor $\frac{3}{2}$ in the exponent of the complexity analysis of the main Theorem 1:

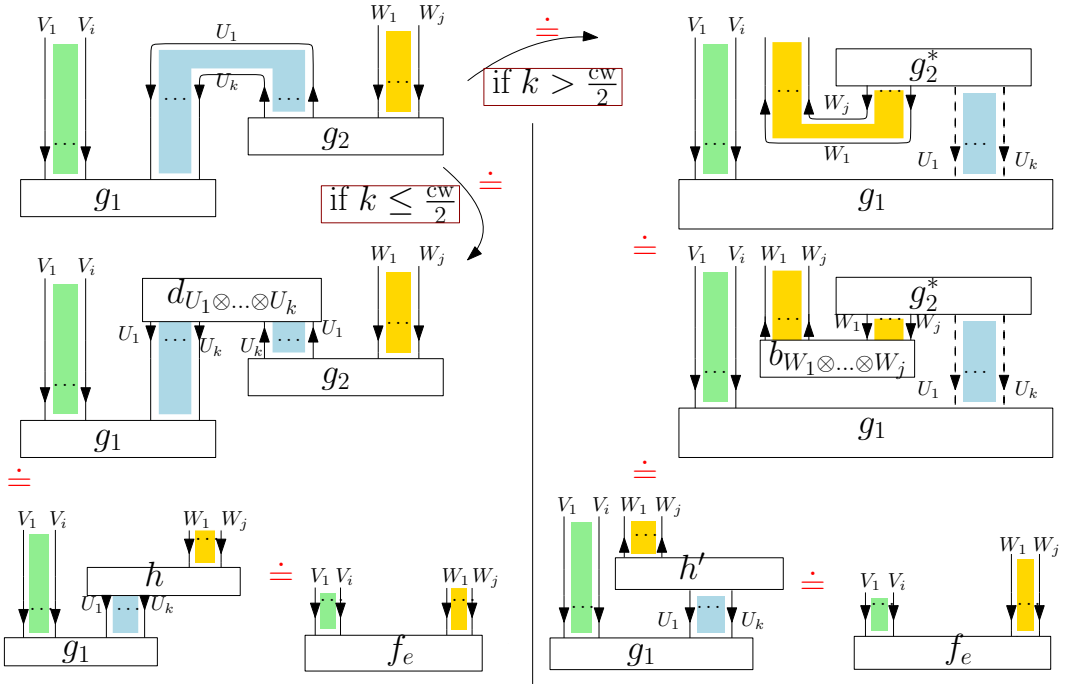


Figure 8 Merging of two coupons in a canonical form (top left) along k strands coloured $U_1, \ldots, U_k$. The factorisation scheme differs whether $k \leq \mathrm{cw}/2$ (left column) or $k > \mathrm{cw}/2$ (right column). The top right equivalence comes from the equality in Figure 9.

Small bridge. For k at most half the carving-width (Figure 8, Left), we consider first the morphism $d_{U_1 \otimes \ldots \otimes U_k}$ induced by the composition of the evaluation morphisms d_{U_ℓ}, $\ell = k \ldots 1$. More precisely, the morphism $d_{U_1 \otimes \ldots \otimes U_k} \colon U_1 \otimes \ldots \otimes U_k \otimes U_k^* \otimes \ldots \otimes U_1^* \to \mathbb{1}$, is obtained by composing the evaluation morphisms from bottom up:

$$\begin{aligned} d_{U_1 \otimes \ldots \otimes U_k} \quad &: \quad U_1 \otimes \ldots \otimes U_k \otimes U_k^* \otimes \ldots \otimes U_1^* \to \mathbb{1}, \\ &= \quad \prod_{\ell=k}^{1} \left(\mathrm{id}_{U_1 \otimes \ldots \otimes U_{\ell-1}} \otimes d_{U_\ell^*} \otimes \mathrm{id}_{U_{\ell-1}^* \otimes \ldots \otimes U_1^*} \right), \end{aligned} \tag{5}$$

where $\ell = k$ is the rightmost term of the composition.

The (partial) composition of $d_{U_1 \otimes \ldots \otimes U_k}$ with g_2 through $U_k^* \otimes \ldots \otimes U_1^*$ gives the morphism:

$$h \colon U_1 \otimes \ldots \otimes U_k \to W_1 \otimes \ldots \otimes W_j, \qquad h := \left(d_{U_1 \otimes \ldots \otimes U_k} \otimes \mathrm{id}_{W_1 \otimes \ldots \otimes W_j} \right) \circ \left(\mathrm{id}_{U_1 \otimes \ldots \otimes U_k} \otimes g_2 \right). \tag{6}$$

Finally, the morphism f_e obtained from the merging of f_{e_1} and f_{e_2} is given by the (partial) composition of g_1 and h, through $U_1 \otimes \ldots \otimes U_k$. Precisely,

$$f_e \colon \mathbb{1} \to V_1 \otimes \ldots \otimes V_i \otimes W_1 \otimes \ldots \otimes W_j, \qquad f_e := \left(\mathrm{id}_{V_1 \otimes \ldots \otimes V_i} \otimes h \right) \circ g_1. \tag{7}$$

By construction, these operations give the morphism f_e induced by graphical calculus on the coloured tangle associated to the subtree of $\mathcal{T}$ rooted at the child node of edge e.

Large bridge. The case k strictly larger than half the carving-width starts by flipping upside-down coupon g_2. This operation is depicted in Figure 9. Starting with a morphism g, it consists of a planar isotopy to produce g^*, the dual morphism to g. In the case where the category $\mathcal{C}$ satisfies the hypothesis of Theorem 2, Figure 9, depicting an isotopy, proves the equality: $(d_U \otimes \mathrm{id}_V) \circ (\mathrm{id}_{U^*} \otimes g) = (\mathrm{id}_V \otimes g^*) \circ (b_V \otimes \mathrm{id}_{U^*})$.

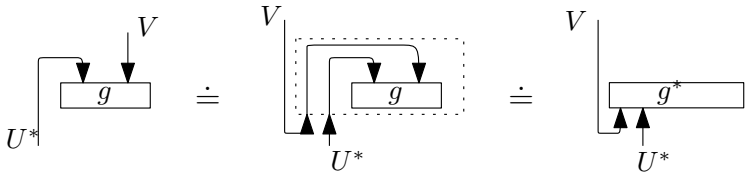


Figure 9 Planar isotopy, then factorisation with g^*, the dual morphism to g.

Applied to the canonical form on g_1 and g_2 (Figure 8, Top) the operation gives the composition of morphisms, involving g_1 and g_2^*, in the top right corner of Figure 8.

The following compositions are similar to the case of a small bridge. Morphism $b_{W_1\otimes\ldots\otimes W_j}$ describes the composition of the co-evaluation morphisms for $W_1,\ldots,W_j$, *i.e.*,

$$\begin{aligned} b_{W_1\otimes\ldots\otimes W_j} &: \mathbb{1} \to W_1^*\otimes\ldots\otimes W_j^*\otimes W_j\otimes\ldots\otimes W_1, \\ &= \prod_{\ell=1}^{j}\left(\mathrm{id}_{W_1^*\otimes\ldots\otimes W_{\ell-1}^*}\otimes b_{V_\ell}\otimes \mathrm{id}_{W_{\ell-1}\otimes\ldots\otimes W_1}\right), \end{aligned} \tag{8}$$

where $\ell = 1$ is the rightmost term of the composition.

The morphism h' is obtained by (partial) composition of $b_{W_1\otimes\ldots\otimes W_j}$ and g_2^*: $h' = \left(\mathrm{id}_{W_1^*\otimes\ldots\otimes W_j^*}\otimes g_2^*\right)\circ\left(b_{W_1\otimes\ldots\otimes W_j}\otimes \mathrm{id}_{U_1\otimes\ldots\otimes U_k}\right)$, and f_e is obtained by (partial) composition of g_1 and h': $f_e = (\mathrm{id}_{V_1\otimes\ldots\otimes V_i}\otimes h')\circ g_1$.

Correctness. The algorithm consists uniquely of:

- ambient isotopies, when ordering tangle endpoints (Figure 6), sliding strands under coupons (tree leaves Figure 5, and inner tree nodes Figure 7), and flipping coupons upside down (dual morphism, Figure 9), and,
- factorisations of morphisms in a specific order (Figure 8).

By virtue of Theorem 2, ambient isotopies do not affect the output of the algorithm (even though it changes computation), as the algorithm computes a topological invariant of links. By properties of ribbon categories, and more specifically the equivalence relations pictured in Figure 2, the order of factorisation of morphisms does not change the output. In conclusion, the high-level algorithm presented in Sections 3 and 4 outputs the topological invariant $J_L^{\mathcal{C}}(V_1,\ldots,V_m)$ for a $(V_1,\ldots,V_m)$-coloured link diagram L.

5 Algebraic implementation and complexity

For the implementation of the algorithm, we assume that the objects in the category $\mathcal{C}$ are finite dimensional free $\mathcal{R}$-modules, for a commutative ring with unity $\mathcal{R}$ and usual tensor product. Denote the dimension of every link component colour V_i by $N_i := \dim V_i$, and let $N := \max_i\{\dim V_i\}$. Fixing a basis for every V_i, all morphisms in $\mathcal{C}$ – in particular the distinguished braiding, evaluation and co-evaluation, and twist morphisms – are represented by matrices with $\mathcal{R}$ coefficients.

This model is general, and contains all quantum invariants derived from simple Lie algebras. We describe seven elementary matrix operations in Section 5.1, and use them in Section 5.2 as building blocks to implement and analyse the complexity of the algorithm for computing quantum invariants.

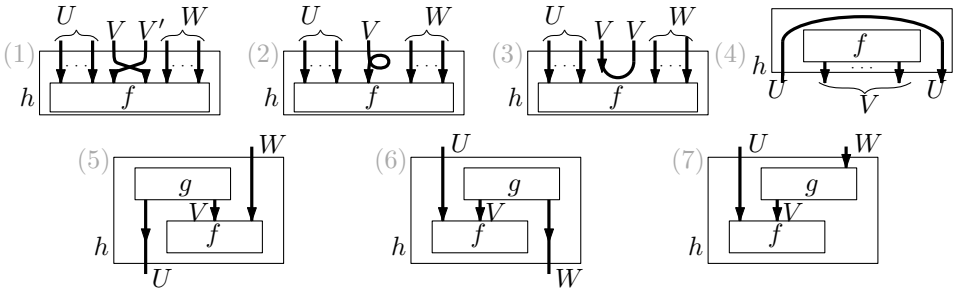


Figure 10 Graphical representation of the seven elementary compositions of morphisms.

5.1 Elementary compositions

We consider the seven elementary compositions of morphisms depicted in Figure 10. They respectively represent the composition with (1) a single braiding, (2) a single twist, (3) a single co-evaluation, (4) a single evaluation. Cases (5), (6), and (7) represent three types of partial compositions of the morphisms f and g. We describe algorithms to perform these compositions on matrices, then use them in Section 5.2 to implement the FPT algorithm described in Sections 3 and 4.

▶ **Lemma 6.** *Consider the elementary morphism compositions in Figure 10 (1), (2), (3), and (4). Let U, V, V', W be finite dimensional free $\mathcal{R}$-modules, with $\dim U = a$, $\dim V = b$, $\dim V' = b'$, and $\dim W = c$. Then, given the matrices for morphisms f, $\theta_V^{\pm}$, $c_{V,V'}^{\pm}$, b_V, and d_U, we can compute the matrix for morphism h in:*

- *$O(a(bb')^2c)$ arithmetic operations in $\mathcal{R}$ for (1),*
- *$O(abc)$ arithmetic operations for (2),*
- *$O(ab^2c)$ arithmetic operations for (3), and*
- *$O(a^2b)$ arithmetic operations for (4).*

The memory complexity of the operation does not exceed the size of the output, which is a row or column vector h containing scalars from $\mathcal{R}$.

Proof.

Figure 10 (1), (2), and (3). All three cases consist of the matrix-vector product $h = (\mathrm{id}_U \otimes M \otimes \mathrm{id}_W) \cdot f$, where M is respectively the $(bb' \times bb')$-matrix $c_{V,V'}^{\pm}$, the $(b \times b)$-matrix $\theta_V^{\pm}$, and the $(b^2 \times 1)$-matrix b_V. Consider M to be an $(m \times m')$-matrix, with coefficients $(M_{i,j})_{1 \le i \le m, 1 \le j \le m'}$. Matrix $(\mathrm{id}_U \otimes M \otimes \mathrm{id}_W)$ has at most m' non-zero coefficients per row. Restricting to these non-zero coefficients, we get the i^{th} entry of h:

$$h_{i,1} = \sum_{k=1\ldots m'} M_{\beta+1,k} \cdot f_{\alpha cm' + \gamma + (k-1)c, 1},$$

where i is uniquely written as $i = \alpha \cdot cm + \beta \cdot c + \gamma$, with $0 \le \alpha \le a-1$, $0 \le \beta \le m-1$, and $1 \le \gamma \le c$. Computing h requires $O(m'|f|)$ arithmetic operations in $\mathcal{R}$, where $|f|$ is the length of vector f, storing $O(|f|)$ scalars from $\mathcal{R}$. Note that matrix $(\mathrm{id}_U \otimes M \otimes \mathrm{id}_W)$ does not need to be explicitly constructed, and only matrix M suffices.

Figure 10 (4) is treated similarly; see [23] for an explicit formula. ◀

▶ **Lemma 7.** *Consider the elementary morphism compositions in Figure 10 (5), (6), and (7). Let U, V, W be finite dimensional free $\mathcal{R}$-modules, with $\dim U = a$, $\dim V = b$, and $\dim W = c$. Then, given the matrices for morphisms f and g, we can compute the matrix for morphism h in $O(abc)$ arithmetic operations in $\mathcal{R}$, and memory complexity $O(ab + bc + ac)$ times the size of a scalar in $\mathcal{R}$.*

Proof. In the same spirit of Lemma 6, the proof exploits the sparsity of tensoring with the identity. Explicit formulae can be found in [23]. ◀

5.2 Implementation and complexity analysis

We implement the algorithm described in Sections 3 and 4 using the elementary compositions of Figure 10. Let N be an upper bound on the dimensions of the different modules U_i, V_j, W_k, over all i, j, k, colouring the components of the link.

Leaf morphisms. The leaf morphisms described in Equations (1)-(4) and Figure 5 are implemented using elementary compositions (1) and (2). By Lemma 6, the complexity is at most $O(N^6)$ arithmetic operations in $\mathcal{R}$.

Sliding under a coupon. The sliding operation as presented in Figure 7 composes a morphism f with a sequence of twist and braiding morphisms. More precisely, let h denote the entire morphism in Figure 7. Starting from the $(O(N^{i+j}) \times 1)$ matrix f, it is computed iteratively applying j times elementary composition (2) for the twists, then $j(j-1)$ times elementary composition (1) for the braidings between V_i- and V_j-strands, and finally ij times elementary composition (1) for the braidings between V_i- and U_j-strands.

During the computation, we maintain a vector of size $(1 \times O(N^{i+j}))$. Applying Lemma 6, the sliding operation runs in $O(j(i+j)N^{i+j+2})$ arithmetic operations in $\mathcal{R}$, storing $O(N^{i+j})$ scalars from $\mathcal{R}$. In the algorithm, $i + j \leq \mathrm{cw}$, the carving-width of the link diagram. Consequently, we get $O(\mathrm{cw}^2\, N^{\mathrm{cw}+2})$ operations, with memory $O(N^{\mathrm{cw}})$.

Construction of evaluations and co-evaluations. The morphism $d_{U_1 \otimes \ldots \otimes U_k}$ appearing in Figure 8 is the result of k elementary compositions of type (4). The morphisms maintained during the computation are of size $(1 \times O(N^{2k}))$. Applying Lemma 6, the computation takes a total of $O(kN^{2k})$ arithmetic operations in $\mathcal{R}$, storing $O(N^{2k})$ scalars from $\mathcal{R}$. The case $b_{W_1 \otimes \ldots \otimes W_j}$ is similar. In the algorithm, k (or j) is at most $\mathrm{cw}/2$. Consequently, the complexity is $O(\mathrm{cw}\, N^{\mathrm{cw}})$ arithmetic operations, storing $O(N^{\mathrm{cw}})$ scalars.

Composition of morphisms. Finally, the compositions of morphisms described in Figure 8 are implemented with a constant number of elementary compositions (5), (6), and (7). Considering Lemma 7, the product abc of dimensions never exceeds $N^{\frac{3}{2}\,\mathrm{cw}}$. Indeed, by property of the tree embedding, we have $i + k, j + k, i + j \leq \mathrm{cw}$ in Figure 8 ; the dichotomy with $k \overset{?}{\leq} \mathrm{cw}/2$ further ensures than no more than $3/2\,\mathrm{cw}$ are involved in the elementary compositions. Consequently, the compositions of Figure 8 are implemented using $O(N^{\frac{3}{2}\,\mathrm{cw}})$ arithmetic operations in $\mathcal{R}$, storing $O(N^{\mathrm{cw}})$ scalars from $\mathcal{R}$.

Overall complexity. In conclusion, we sum up the different steps of the algorithm and its implementation. Let D be a coloured link diagram with n crossings and carving width cw, where the dimension of each colouring module is at most N. The algorithm first computes an optimal tree embedding in $O(\mathrm{poly}(n))$ operations. The tree has size n and width cw. W.l.o.g., we assume the diagram has at least one crossing that is not a twist, and consequently $\mathrm{cw} \geq 4$, the maximal degree of the graph. Considering $\mathrm{cw} \in O(\sqrt{n})$ and $4 \leq \mathrm{cw}+2 \leq \frac{3}{2}\,\mathrm{cw}$, the quantum invariant associated to the colouring is computed in:

$$O(nN^{\frac{3}{2}\,\mathrm{cw}} + n^2 N^{\mathrm{cw}+2}) \text{ arithmetic operations in } \mathcal{R},$$

storing: $O(n)$ words for the diagram, plus $O(N^{\mathrm{cw}})$ scalars from $\mathcal{R}$ for the matrices.

Note that this complexity is expressed in *algebraic operations in* $\mathcal{R}$, when the main Theorem 1 is expressed in *machine operations*. We bound the arithmetic complexity of operations in $\mathcal{R}$ in the following Section 6.

6 Bounding arithmetic complexity and proof of the main theorem

Working with matrices with $\mathcal{R}$-coefficients, for a ring $\mathcal{R}$, allows the algorithm to be applied in great generality. We focus on the case $\mathcal{R} = \mathbb{Z}[q]$, which is sufficient for all $J_L^{\mathfrak{g}}$ invariants of Theorem 1. See [29, Chapter 6] for an explicit construction of the $J_L^{\mathfrak{g}}$.

▶ Remark 8. Note that quantum invariants are usually defined in the category of $\mathbb{Z}[q, q^{-1}]$-modules. Multiplying the braiding, twist, and (co)evaluation matrices by q^a for a large enough, and re-normalising the output, allows us to restrict to the case of $\mathbb{Z}[q]$-modules.

Processing matrix multiplications with coefficients in $\mathbb{Z}[q]$, both degrees of polynomials as well as values of coefficients may blow-up during intermediate computation. Specifically, both arithmetic operations in $\mathcal{R}$ and bit size of $\mathcal{R}$-elements may become *exponential in* n.

In the long version of the article [23], we describe a solution based on Lagrange interpolation and the Chinese remainder theorem to reconstruct polynomial $J_L^{\mathfrak{g}}$ from evaluations, using only integers of bit size $O(\log n)$. Specifically, we prove that, for a fixed category $\mathcal{C}$ of $\mathbb{Z}[q]$-modules:

▶ **Proposition 9.** *The invariant $J_L^{\mathcal{C}}(V_1, \dots, V_m) \in \mathbb{Z}[q]$ of an m-component link L, of size n and coloured with $V_1, \dots, V_m \in \mathcal{C}$ of dimension at most N, can be computed by:*

- *running $O(n^2\sqrt{n})$ times the FPT algorithm described in this article, using matrices with coefficients in $\mathbb{Z}/p\mathbb{Z}$, for various primes $p \in O(n\sqrt{n})$,*
- *and an extra cost of $O(n^4)$ for large integers reconstruction (Chinese remainder theorem) and polynomial reconstruction (Lagrange interpolation).*

The complexity analysis of Section 5.2 and Theorem 9 allows us to conclude the proof of the main Theorem 1. Considering arithmetic operations in $\mathbb{Z}/p\mathbb{Z}$, $p \in O(n\sqrt{n})$, constant time, the algorithm has complexity $O(n^3\sqrt{n}N^{\frac{3}{2}\,\mathrm{cw}} + n^4\sqrt{n}N^{\mathrm{cw}+2})$ *machine operations*.

Conclusion. We have introduced a general sub-exponential, and fixed parameter tractable algorithm to compute any quantum knot invariants derived from a ribbon category, which include infinite families of meaningful knot invariants such as coloured Jones polynomials.

We mention two open research directions:

- This new algorithm offers a new tool to verify important mathematical conjectures [7, 8, 16, 24] on knots that where until now intractable computationally. As a proof of concept, we have implemented the algorithm for coloured Jones polynomials and verified experimentally some behaviours related to the *volume conjecture* and the *slope conjecture* in [23].
- No lower bound is known for the computation of quantum invariants. However, progress [9] on the complexity of *planar Tutte polynomials* under the *exponential time hypothesis* (ETH) may lead to a lower bound of $2^{\Omega(\sqrt{n})}$ for the Jones polynomial, which would imply that our FPT algorithm is optimal up to a constant in the exponent, under ETH.

References

1 Dorit Aharonov, Vaughan Jones, and Zeph Landau. A polynomial quantum algorithm for approximating the jones polynomial. *Algorithmica*, 55:395–421, 2009.

2 Daniel Bienstock. On embedding graphs in trees. *Journal of Combinatorial Theory, Ser. B*, 49(1):103–136, 1990. doi:10.1016/0095-8956(90)90066-9.

3 Benjamin A. Burton. The HOMFLY-PT polynomial is fixed-parameter tractable. In *34th International Symposium on Computational Geometry, SoCG 2018*, pages 18:1–18:14, 2018. doi:10.4230/LIPIcs.SoCG.2018.18.

4 Benjamin A. Burton. The next 350 million knots. In *36th International Symposium on Computational Geometry, SoCG 2020*, pages 25:1–25:17, 2020.

5 Benjamin A. Burton, Clément Maria, and Jonathan Spreer. Algorithms and complexity for Turaev-Viro invariants. *Journal of Applied and Computational Topology*, 2(1-2):33–53, 2018. doi:10.1007/s41468-018-0016-2.

6 Vladimir G. Drinfeld. Quantum groups. In *Proceedings International Congress of Mathematicians*, pages 798–820, 1986.

7 Stavros Garoufalidis. On the characteristic and deformation varieties of a knot. *Geometry & Topology Monographs*, 7:291–309, 2004.

8 Stavros Garoufalidis. The Jones slopes of a knot. *Quantum Topology*, 2(1):43–69, 2011.

9 Leslie Ann Goldberg and Mark Jerrum. Inapproximability of the Tutte polynomial of a planar graph. *Computational Complexity*, 21:605–642, 2012.

10 Qian-Ping Gu and Hisao Tamaki. Optimal branch-decomposition of planar graphs in O(n3) time. *ACM Transaction on Algorithms*, 4(3):30:1–30:13, 2008.

11 Kristóf Huszár and Jonathan Spreer. 3-manifold triangulations with small treewidth. In *35th International Symposium on Computational Geometry, SoCG 2019*, pages 44:1–44:20, 2019. doi:10.4230/LIPIcs.SoCG.2019.44.

12 Kristóf Huszár, Jonathan Spreer, and Uli Wagner. On the treewidth of triangulated 3-manifolds. *Journal of Computational Geometry*, 2(10):70–98, 2019.

13 François Jaeger, Dirk L. Vertigan, and Dominic J. A. Welsh. On the computational complexity of the Jones and Tutte polynomials. *Mathematical Proceedings of the Cambridge Philosophical Society*, 108(1):35–53, 1990.

14 Michio Jimbo. A q-difference analogue of U(g) and the Yang-Baxter equation. *Letters in Mathematical Physics*, 10(1):63–69, 1985.

15 Vaughan F. R. Jones. A polynomial invariant for knots via von Neumann algebras. *Bulletin of the American Mathematical Society*, 12:103–111, 1985.

16 Rinat M. Kashaev. The hyperbolic volume of knots from the quantum dilogarithm. *Letters in Mathematical Physics*, 39(3):269–275, 1997.

17 W. B. Raymond Lickorish. *An Introduction to Knot Theory*. Graduate Texts in Mathematics. Springer-Verlag New York, 1997.

18 Richard J. Lipton and Robert E. Tarjan. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics*, 36(2):177–189, 1979.

19 Janos A. Makowsky and Julián P. Mariño. The parameterized complexity of knot polynomials. *Journal of Computer and System Sciences*, 67(4):742–756, 2003.

20 Clément Maria and Jessica S. Purcell. Treewidth, crushing, and hyperbolic volume. *Algebraic & Geometric Topology*, 19:2625–2652, 2019.

21 Clément Maria and Owen Rouillé. Computation of large r asymptotics of 3-manifold quantum invariants. In *Proceedings of the 23rd Meeting on Algorithm Engineering and Experiments, ALENEX 2021*. SIAM, 2021.

22 Clément Maria and Jonathan Spreer. A polynomial-time algorithm to compute Turaev-Viro invariants TV4,q of 3-manifolds with bounded first Betti number. *Found. Comput. Math.*, 20(5):1013–1034, 2020. doi:10.1007/s10208-019-09438-8.

23 Clément Maria. Parameterized complexity of quantum knot invariants, 2019. arXiv:1910.00477.

24 Hitoshi Murakami and Jun Murakami. The colored Jones polynomials and the simplicial volume of a knot. *Acta Mathematica*, 186(1):85–104, 2001.

25 Nicolai Y. Reshetikhin and Vladimir G. Turaev. Ribbon graphs and their invariants derived from quantum groups. *Communications in Mathematical Physics*, 127(1):1–26, 1990.

26 Neil Robertson and Paul D. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *Journal of Algorithms*, 7(3):309–322, 1986.

27 Saul Schleimer, Arnaud de Mesmay, Jessica Purcell, and Eric Sedgwick. On the tree-width of knot diagrams. *Journal of Computational Geometry*, 10(1):164–180, 2019.

28 Paul D. Seymour and Robin Thomas. Call routing and the ratcatcher. *Combinatorica*, 14(2):217–241, 1994.

29 Vladimir G. Turaev. *Quantum Invariants of Knots and 3-Manifolds*. de Gruyter Studies in Mathematics. Walter de Gruyter & Co., Berlin, revised edition, 2010.

30 Vladimir G. Turaev and Oleg Y. Viro. State sum invariants of 3-manifolds and quantum $6j$-symbols. *Topology*, 31(4):865–902, 1992.

Efficient Generation of Rectangulations via Permutation Languages

Arturo Merino
TU Berlin, Germany

Torsten Mütze
University of Warwick, Coventry, United Kingdom
Charles University, Prague, Czech Republic

Abstract

A generic rectangulation is a partition of a rectangle into finitely many interior-disjoint rectangles, such that no four rectangles meet in a point. In this work we present a versatile algorithmic framework for exhaustively generating a large variety of different classes of generic rectangulations. Our algorithms work under very mild assumptions, and apply to a large number of rectangulation classes known from the literature, such as generic rectangulations, diagonal rectangulations, 1-sided/area-universal, block-aligned rectangulations, and their guillotine variants. They also apply to classes of rectangulations that are characterized by avoiding certain patterns, and in this work we initiate a systematic investigation of pattern avoidance in rectangulations. Our generation algorithms are efficient, in some cases even loopless or constant amortized time, i.e., each new rectangulation is generated in constant time in the worst case or on average, respectively. Moreover, the Gray codes we obtain are cyclic, and sometimes provably optimal, in the sense that they correspond to a Hamilton cycle on the skeleton of an underlying polytope. These results are obtained by encoding rectangulations as permutations, and by applying our recently developed permutation language framework.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Mathematics of computing → Discrete mathematics

Keywords and phrases Exhaustive generation, Gray code, flip graph, polytope, generic rectangulation, diagonal rectangulation, cartogram, floorplan, permutation pattern

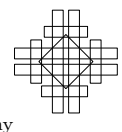
Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.54

Related Version *Full Version*: `arXiv:2103.09333` [23]

Funding This work was supported by German Science Foundation grant 413902284.
Arturo Merino: Supported by ANID Becas Chile 2019-72200522.
Torsten Mütze: Supported by Czech Science Foundation grant GA 19-08554S.

1 Introduction

Partitioning a geometric shape into smaller shapes is a fundamental theme in discrete and combinatorial geometry. In this paper we consider *rectangulations*, i.e., partitions of a rectangle into finitely many interior-disjoint rectangles. Such partitions have an abundance of practical applications, which motivates their combinatorial and algorithmic study. For example, rectangulations are an appealing way to represent geographic information as a cartogram. This is a map where each country is represented as a rectangle, the adjacencies between rectangles correspond to those between countries, and the areas of the rectangles are determined by some geographic variable, such as population size [34]. If the rectangulation is *area-universal* [12], then such an adjacency-preserving cartogram can be drawn for any assignment of area values to the rectangles. Another important use of rectangulations is as floorplans in VLSI design and architectural design. These problems often involve additional



constraints on top of adjacency, such as extra space for wires [27] or proportion limits for the rooms [24]. An important notion in this context are *slicing* floorplans [27], i.e., floorplans that can be subdivided into rectangles by a sequence of vertical or horizontal guillotine cuts.

Rectangulations have rich combinatorial properties, and a task that has received a lot of attention is counting, i.e., determining the number of rectangulations of a particular type with n rectangles, either exactly as a function of n [37] or asymptotically as n grows [32]. This led to several beautiful bijections of rectangulations with pattern-avoiding permutations [1, 4, 29] or with twin binary trees [37]. The focus of this paper is on another fundamental algorithmic task, which is more fine-grained than counting, namely exhaustive generation, meaning that every rectangulation from a given class must be produced exactly once. While such generation algorithms are known for many other discrete objects such as permutations, combinations, subsets, trees etc. and covered in standard textbooks such as Knuth's [20], much less is known about the generation of geometric objects such as rectangulations.

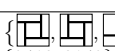
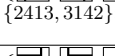
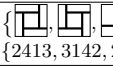
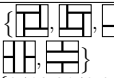
The ultimate goal for a generation algorithm is to produce each new object in time $\mathcal{O}(1)$, which requires that consecutively generated objects differ only by a "small local change". Such a minimum change listing of combinatorial objects is often called a *Gray code* [31]. If the time bound $\mathcal{O}(1)$ for producing the next object holds in every step, then the algorithm is called *loopless* [11], and if it holds on average it is called *constant amortized time* (CAT) [30]. The Gray code problem entails the definition of a *flip graph*, which has as nodes all the combinatorial objects to be generated, and an edge between any two objects that differ in the specified small way. Clearly, computing a Gray code ordering of the objects is equivalent to traversing a Hamilton path or cycle in the corresponding flip graph. It turns out that some interesting flip graphs arising from rectangulations can be equipped with a natural lattice structure [21, 22], analogous to the Tamari lattice on triangulations, and realized as polytopes in high-dimensional space [28], analogous to the associahedron. This ties in the Gray code problem with deep methods and results from lattice and polytope theory.

1.1 Our results

The main contribution of this paper is a versatile algorithmic framework for generating a large variety of different classes of generic rectangulations, i.e., rectangulations with the property that no four rectangles meet in a point. In particular, we obtain efficient generation algorithms for several interesting classes known from the literature, in some cases loopless or CAT algorithms; see Table 1. The initialization time and memory requirement for all these algorithms is linear in the number of rectangles. The classes of rectangulations shown in the table arise from generic rectangulations by imposing structural constraints, such as the guillotine property or forbidden configurations, or by equivalence relations, and they will be defined in Section 2.2. We implemented the algorithms generating the classes of rectangulations from the table in C++, and we made the code available for download and experimentation on the Combinatorial Object Server [10].

The classes of rectangulations that our algorithms can generate are not limited to the examples shown in Table 1, but can be described by the following *closure property*; see Figure 1. Given an infinite class of rectangulations $\mathcal{C}$, we require that if a rectangulation R is contained in $\mathcal{C}$, then the rectangulation obtained from R by deleting the bottom-right rectangle is also in $\mathcal{C}$, and the two rectangulations obtained from R by inserting a new rectangle at the bottom or right, respectively, are also in $\mathcal{C}$. If $\mathcal{C}$ satisfies this property, then our algorithms allow generating the set $\mathcal{C}_n \subseteq \mathcal{C}$ of all rectangulations from $\mathcal{C}$ with exactly n rectangles, for every $n \geq 1$, by so-called *jumps*, a minimum change operation that generalizes simple flips, T-flips, and wall slides studied in [8, 29]. Moreover, if the class $\mathcal{C}$ is symmetric,

Table 1 Classes of rectangulations that can be generated by our algorithms. The second column gives a description of the class in terms of forbidden rectangulation patterns (n/a means not applicable), and one or more bijectively equivalent classes of pattern-avoiding permutations. Underlined and overlined permutation patterns are so-called vincular and barred patterns; see [15] and the papers referenced in the table. The last column specifies the obtained runtime bound for generating each rectangulation, where n is the number of rectangles. These are all worst case bounds that apply in every step (in particular, LL=loopless), with the exception of the $\mathcal{O}(1)$ bound for generic rectangulations, which holds on average (CAT=constant amortized time). For more extensive counting results on pattern-avoiding rectangulations, see [23].

Class		**Forbidden patterns**	**Counts/OEIS** [26]	**Refs.**	**Runtime**
generic		$\emptyset$ $\{3\underline{5}124, 3\underline{5}142, 24\underline{5}13, 42\underline{5}13\}$: 2-clumped permutations	$1, 2, 6, 24, 116, 642, 3938, 26194, \ldots$	[22, 29]	$\mathcal{O}(1)$ CAT
diagonal =mosaic floorpl. /R-equivalence		$\{$  $\}$ $\{2\underline{4}13, 3\underline{1}42\}$: Baxter $\{2\underline{4}13, 3\underline{4}12\}$: twisted Baxter $\{2\underline{1}43, 3\underline{1}42\}$	$1, 2, 6, 22, 92, 422, 2074, 10754, \ldots$ A001181 (Baxter numbers)	[1, 8] [21, 37]	$\mathcal{O}(1)$ LL
1-sided =area-universal		$\{$  $\}$	$1, 2, 6, 20, 72, 274, 1088, 4470, \ldots$	[12]	$\mathcal{O}(n)$
block-aligned /S-equivalence		n/a $\{2\underline{1}43, 3\underline{4}12\}$	$1, 1, 2, 6, 22, 88, 374, 1668, 7744, \ldots$ A214358	[4]	$\mathcal{O}(1)$ LL
guillotine	generic	$\{$  $\}$	$1, 2, 6, 24, 114, 606, 3494, 21434, \ldots$		$\mathcal{O}(n)$
	diagonal =slicing fl.pl. /R-equiv.	$\{$  $\}$ $\{2413, 3142\}$: separable	$1, 2, 6, 22, 90, 394, 1806, 8558, \ldots$ A006318 (Schröder numbers)	[1, 37] [4, 5]	$\mathcal{O}(n)$
	1-sided	$\{$  $\}$ $\{2413, 3142, 2\bar{1}\bar{3}54, 4\bar{5}\bar{3}12\}$	$1, 2, 6, 20, 70, 254, 948, 3618, \ldots$ A078482	[5]	$\mathcal{O}(n)$
		$\{$  $\}$ {2413,3142,2143,3412}	$1, 2, 6, 20, 68, 232, 792, 2704, \ldots$ A006012	[5]	$\mathcal{O}(n^2)$
	block-aligned /S-equiv.	n/a $\{2413, 3142, 2\underline{1}43, 3\underline{4}12\}$	$1, 1, 2, 6, 20, 70, 254, 948, 3618, \ldots$ A078482	[4]	$\mathcal{O}(n)$

i.e., if R is in $\mathcal{C}$ then the rectangulation obtained from R by reflection at the diagonal from top-left to bottom-right is also in $\mathcal{C}$, then the jump Gray code for $\mathcal{C}_n$ is cyclic, i.e., the last rectangulation differs from the first one only by a jump. In other words, we not only obtain a Hamilton path in the corresponding flip graph, but a Hamilton cycle. In fact, all the classes of rectangulations listed in Table 1 satisfy the aforementioned closure and symmetry properties, so in all those cases we obtain cyclic jump Gray codes.

Generic rectangulations and diagonal rectangulations, shown in the first two rows of Table 1, have an underlying lattice and polytope structure [21, 22, 28], and in those two cases our Gray codes form a Hamilton cycle on the skeleton of this polytope, i.e., jumps are provably optimal minimum change operations.

It turns out that many interesting classes of rectangulations can be characterized by pattern avoidance; see the second column in Table 1. Under very mild conditions on the patterns, these classes satisfy the aforementioned closure property, and can hence be generated by our framework. In this work we initiate a systematic investigation of pattern avoidance in rectangulations, and we obtain the first counting results for many known and new classes; see the third column in Table 1 and the more extensive tables in [23].

Our generation framework for rectangulations consists of two main algorithms. The first is a simple greedy algorithm that generates a jump Gray code ordering for any set of rectangulations $\mathcal{C}_n \subseteq \mathcal{C}$ for which $\mathcal{C}$ satisfies the aforementioned closure property; see

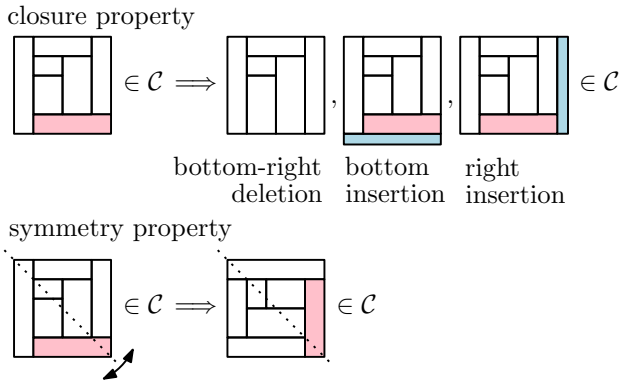


Figure 1 Closure property and symmetry property.

Algorithm $\mathrm{J}^{\square}$ and Theorem 5 in Section 3. The second is a memoryless version of the first algorithm, which computes the same ordering of rectangulations; see Algorithm $\mathrm{M}^{\square}$ and Theorem 8 in Section 5. This algorithm can be fine-tuned to derive efficient algorithms for several known rectangulation classes such as the ones listed in Table 1, by providing corresponding jump oracles for the class $\mathcal{C}$.

To prove Theorems 5 and 8, we encode rectangulations by permutations as described by Reading [29], and we then apply our framework for exhaustively generating permutation languages presented in [14, 15, 17]. The minimum change operations on permutations used in that framework translate to jumps on rectangulations. Generating different classes of rectangulations efficiently is thus another major new application of our permutation language framework, and in this paper we flesh out the details of this application.

1.2 Related work

There has been some prior work on generating a few special classes of rectangulations, all based on Avis and Fukuda's reverse search method [6]. Specifically, Nakano [25] described a CAT generation algorithm for generic rectangulations, which does not produce a Gray code, however. This algorithm has been adapted by Takagi and Nakano [33] to generate generic rectangulations with bounds on the number of rectangles that do not touch the outer face. Yoshii, Chigira, Yamanaka and Nakano [38] gave a Gray code for diagonal rectangulations based on a generating tree that is different from ours, resulting in a loopless algorithm. Their Gray code changes at most 3 edges of the rectangulation in each step, whereas our algorithm changes only 1 edge in each step for diagonal rectangulations and generic rectangulations. Consequently, none of the listings produced by these earlier algorithms corresponds to a walk along the skeleton of the underlying polytope.

There has been a lot of work on combinatorial properties of rectangulations. Yao, Chen, Cheng and Graham [37] showed that diagonal rectangulations are counted by the Baxter numbers and that guillotine diagonal rectangulations are counted by the Schröder numbers, using a bijection between diagonal rectangulations and twin binary trees. Ackerman, Barequet and Pinter [1] presented another bijection between diagonal rectangulations and Baxter permutations, which also yields a bijection between guillotine diagonal rectangulations and separable permutations. Shen and Chu [32] provided asymptotic estimates for these two rectangulation classes. Moreover, He [16] presented an optimal encoding of diagonal rectangulations with n rectangles using only $3n - 3$ bits, which is optimal.

The term "generic rectangulation" was coined by Reading [29], who established a bijection between generic rectangulations and 2-clumped permutations, proving that these permutations are representatives of equivalence classes of a lattice congruence of the weak order on the symmetric group. Earlier, generic rectangulations had been studied under the name "rectangular drawings" by Amano, Nakano and Yamanaka [3] and by Inoue, Takahashi and Fujimaki [13, 19], who established recursion formulas and asymptotic bounds for their number. More general classes of rectangular partitions were analyzed by Conant and Michaels [9].

Ackerman, Barequet and Pinter [2] considered the setting where we are given a set of n points in general position in a rectangle, and the goal is to partition the rectangle into smaller rectangles by n walls, such that each point from the set lies on a distinct wall. They showed that for every set of points that forms a separable permutation in the plane, the number of possible rectangulations is the $(n+1)$st Baxter number, and for every point set the number of possible guillotine rectangulations is the nth Schröder number. They also presented a counting and generation procedure based on simple flips and T-flips using reverse search, which was later improved by Yamanaka, Rahman and Nakano [36].

1.3 Outline of this paper

In Section 2 we provide basic definitions and concepts that will be used throughout the paper. In Section 3 we present a greedy algorithm for generating a set of rectangulations by jumps, and we provide a sufficient condition for the algorithm to succeed. In Section 4 we show that the algorithm applies to a large number of rectangulation classes that are characterized by pattern avoidance. In Section 5 we demonstrate how to make our generation algorithm memoryless and efficient. The implementation details of these algorithms and the proofs of Theorems 5 and 8 are omitted due to space constraints; they can be found in [23].

2 Preliminaries

2.1 Generic rectangulations

A *generic rectangulation*, or rectangulation for short, is a partition of a rectangle into finitely many interior-disjoint axis-aligned rectangles, such that no four rectangles of the partition have a point in common; see Figure 2. Given rectangles r and s, we say that r is *left* of s, and s is *right* of r, if the right side of r intersects the left side of s (necessarily in a line segment, rather than a single point). Similarly, we say that r is *below* s, and s is *above* r, if the top side of r intersects the bottom side of s. We consider generic rectangulations up to equivalence that preserves the left/right and below/above relations between rectangles, and we write $\mathcal{R}_n$ for the set of all rectangulations with n rectangles.

We refer to every rectangle corner in a rectangulation as a *vertex*, to every minimal line segment between two vertices as an *edge*, and to every maximal line segment between two vertices that are not corners of the rectangulation as a *wall*. The *type* of a vertex that is not a corner describes the shape of the T-joint at this vertex, and it is one of $\top$, $\vdash$, $\bot$, or $\dashv$.

2.2 Flip operations and classes of rectangulations

Our Gray codes use three types of local change operations on rectangulations; see Figure 3.

A *wall slide* swaps the order of two neighboring vertices of types $\vdash$ and $\dashv$ along a vertical wall, or of types $\top$ and $\bot$ along a horizontal wall. A *simple flip* swaps the orientation of a wall that separates two rectangles. For a vertex v that belongs to three rectangles, consider the wall w that goes through v and the wall t that ends at v, and let w' and w'' be the two halves of w meeting in v. A *T-flip* swaps the orientation of w' or w'' so that it merges with t.

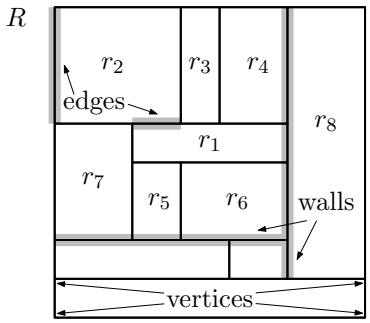


Figure 2 Generic rectangulation R with 11 rectangles. The rectangle r_1 is below r_2, r_3 and r_4, above r_5 and r_6, right of r_7 and left of r_8.

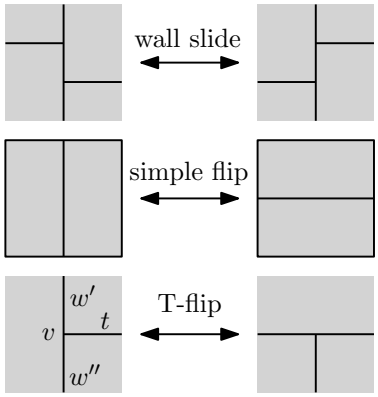


Figure 3 Local change operations on rectangulations.

We now define various interesting subclasses of generic rectangulations that have been studied in the literature and that appear in Table 1. A *diagonal* rectangulation is one in which every rectangle intersects the *main diagonal* that goes from the top-left to the bottom-right corner of the rectangulation. We write $\mathcal{D}_n \subseteq \mathcal{R}_n$ for the set of all diagonal rectangulations with n rectangles. Diagonal rectangulations are characterized by avoiding the wall patterns  and  [8]. Consider the equivalence relation on $\mathcal{R}_n$ obtained from wall slides, sometimes referred to as *R-equivalence* [4]. The equivalence classes are referred to as *mosaic floorplans*, and every equivalence class contains exactly one diagonal rectangulation, obtained by repeatedly destroying occurrences of  or  by wall slides [8]. Consequently, in a diagonal rectangulation, along every vertical wall, all $\vdash$-vertices are below all $\dashv$-vertices, and along every horizontal wall, all $\perp$-vertices are to the left of all $\top$-vertices.

In a *1-sided* rectangulation, every wall is the side of at least one rectangle. This notion was introduced by Eppstein, Mumford, Speckmann, and Verbeek [12], who used it to characterize *area-universal* rectangulations, i.e., for any assignment of areas to the rectangles, the rectangulation can be drawn so that each rectangle has the prescribed area.

Asinowski et al. [4] also considered the equivalence relation on $\mathcal{R}_n$ obtained from wall slides and simple flips, and they called it *S-equivalence*. By definition, S-equivalence is a coarser relation than R-equivalence, i.e., the equivalence classes are obtained by identifying mosaic floorplans that differ in simple flips. In [23] we introduce *block-aligned* rectangulations, which are a subset of diagonal rectangulations with the property that every equivalence class of S-equivalence contains exactly one block-aligned rectangulation.

A rectangulation is *guillotine*, if each of its rectangles can be cut out from the entire rectangulation by a sequence of straight vertical or horizontal cuts. Guillotine rectangulations are characterized by avoiding the windmill patterns  and  . Various special classes of guillotine diagonal rectangulations, characterized by the avoidance of certain wall configurations, were introduced by Asinowski and Mansour [5] (see Section 4 for precise definitions). Mosaic floorplans that are guillotine are also known as *slicing* floorplans.

2.3 Deletion of rectangles

We now describe two operations on a generic rectangulation R with n rectangles, namely deleting a rectangle and inserting a rectangle. The resulting rectangulations have $n-1$ or $n+1$ rectangles, respectively, and they will be denoted by $p(R)$ and $c_i(R)$, notations that refer to the parent and children of R, in a tree structure that will be discussed shortly. The deletion and insertion operations were introduced in [18] and heavily used e.g. in [1] and [25].

The idea of deletion is to contract the rectangle in the bottom-right corner of the rectangulation. Formally, given a rectangulation $R \in \mathcal{R}_n$, $n \geq 2$, we consider the rectangle r in the bottom-right corner, and we consider the top-left vertex of r. If this vertex has type $\vdash$, then we collapse r by sliding its top side, which forms a wall, downwards until it merges with the bottom side of r; see Figure 4 (a). Similarly, if this vertex has type $\top$, then we collapse r by sliding its left side, which forms a wall, to the right until it merges with the right side of r; see Figure 4 (b). We denote the resulting rectangulation with $n-1$ rectangles by $p(R) \in \mathcal{R}_{n-1}$, and we say that $p(R)$ is obtained from R by *deletion*.

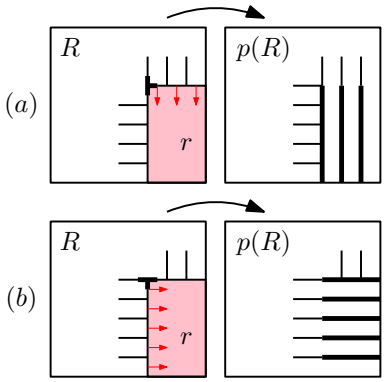


Figure 4 Deletion operation.

Moreover, we denote the n rectangles of R by $r_n, r_{n-1}, \ldots, r_1$ in the order in which they are deleted when applying the deletion operation exhaustively; see Figure 5. Clearly, if r_i is deleted and its top-left vertex has type $\vdash$, then the rightmost rectangle above r_i is r_{i-1}. Similarly, if the top-left vertex has type $\top$, then the lowest rectangle to the left of r_i is r_{i-1}.

For any $R \in \mathcal{R}_n$ and $i = 1, \ldots, n$ we define $R^{[i]} := p^{n-i}(R)$, i.e., this is the subrectangulation of R formed by the first i rectangles; see Figure 5.

2.4 Insertion of rectangles

The idea of insertion is to add a new rectangle into the bottom-right corner of the rectangulation. Given a rectangulation $R \in \mathcal{R}_n$, we first define a set of points in R that can become the top-left corner of the newly added rectangle; see Figure 6.

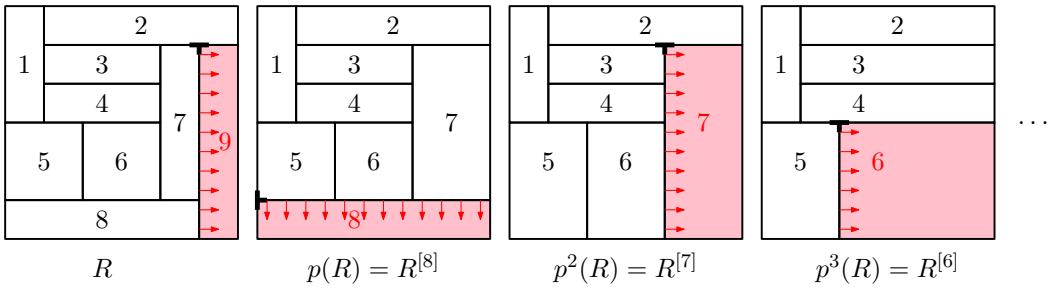


Figure 5 A rectangulation and the indexing of its rectangles given by repeated deletion.

For any rectangle r in $R \in \mathcal{R}_n$, $n \geq 1$, that touches the bottom boundary of R, we consider all edges forming the left side of r, and from every such edge we select one interior point, and we refer to it as a *vertical insertion point*. Similarly, for any rectangle r in R that touches the right boundary of R, we consider the set of all edges forming the top side of r, and from every such edge we select one interior point, and we refer to it as a *horizontal insertion point*. Combinatorially it does not make a difference which interior point is selected.

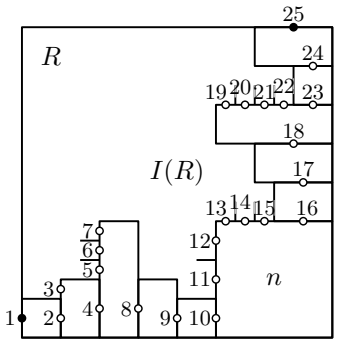


Figure 6 Linear ordering of insertion points. First and last insertion point are filled.

We order the insertion points linearly, by sorting all vertical insertion points lexicographically by their (x, y)-coordinates, followed by all horizontal insertion points sorted lexicographically by their (y, x)-coordinates; see Figure 6. We write $I(R) = (q_1, q_2, \ldots, q_\nu)$ for the sequence of all insertion points ordered in this linear order. In particular, $\nu = \nu(R)$ denotes the number of insertion points.

▶ **Lemma 1.** *For any rectangulation $R \in \mathcal{R}_n$ we have $\nu(R) \leq n + 1$.*

The proof of Lemma 1 is straightforward; see [23]. The upper bound in Lemma 1 is attained if every rectangle touches the bottom or right boundary of R.

Given $R \in \mathcal{R}_n$ and the sequence of insertion points $I(R) = (q_1, \ldots, q_\nu)$, for each $i = 1, \ldots, \nu$ we define a rectangulation $c_i(R) \in \mathcal{R}_{n+1}$ as follows: If q_i is a vertical insertion point, then $c_i(R)$ is obtained from R by inserting a new rectangle r_{n+1} in the bottom-right corner such that r_{n+1} has above it exactly all rectangles which in R lie to the right of q_i and touch the bottom boundary of R, and such that r_{n+1} has to its left exactly all rectangles which in R touch the vertical wall through q_i below q_i; see Figure 7 (a). Similarly, if q_i is a horizontal insertion point, then $c_i(R)$ is obtained from R by inserting a new rectangle r_{n+1} in the bottom-right corner such that r_{n+1} has to its left exactly all rectangles which in R

lie below q_i and touch the right boundary of R, and such that r_{n+1} has above it exactly all rectangles which in R touch the horizontal wall through q_i to the right of q_i; see Figure 7 (b). We say that $c_i(R)$ is obtained from R by *insertion*.

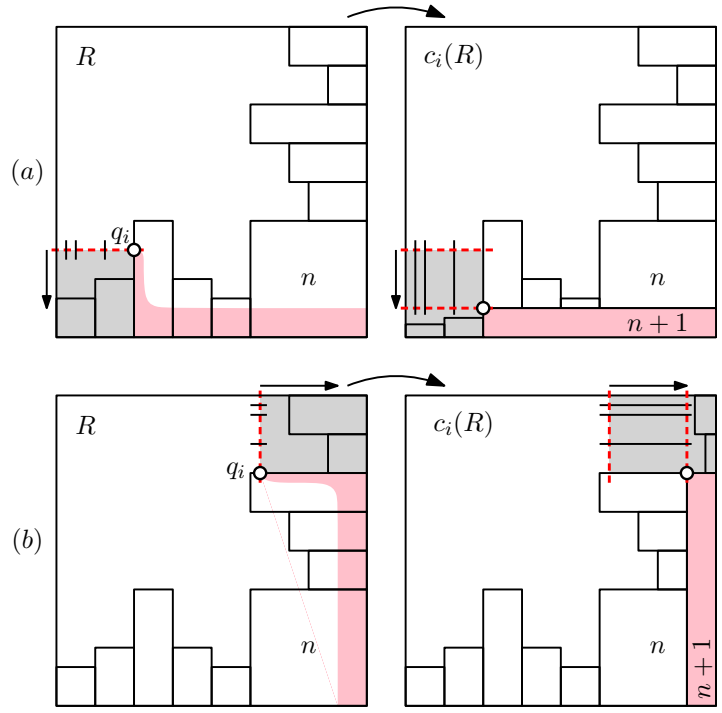


Figure 7 Insertion operation.

By these definitions, the operations of deletion and insertion are inverse to each other, which we record in the following lemma.

▶ **Lemma 2.** *For any rectangulation $R \in \mathcal{R}_n$ and any two distinct insertion points q_i and q_j from $I(R)$, the rectangulations $c_i(R) \in \mathcal{R}_{n+1}$ and $c_j(R) \in \mathcal{R}_{n+1}$ are distinct, and we have $R = p(c_i(R)) = p(c_j(R))$.*

The first and last insertion point, highlighted in Figure 6, play a special role. We say that R is *bottom-based* if R has a rectangle whose bottom side is the entire bottom boundary of R, and R is *right-based* if R has a rectangle whose right side is the entire right boundary of R. Note that $\square$ is both bottom-based and right-based, and if $n \geq 2$, then $R \in \mathcal{R}_n$ is bottom-based if and only if $R = c_1(p(R))$ and right-based if and only if $R = c_{\nu(p(R))}(p(R))$.

3 The basic algorithm

We now present the basic algorithm that we use to generate a set of rectangulations $\mathcal{C}_n \subseteq \mathcal{R}_n$.

3.1 Jumps in rectangulations

We first introduce a local change operation that generalizes the three kinds of flips introduced in Section 2.2 (recall Figure 3) and that will be applied when moving from one rectangulation in $\mathcal{C}_n$ to the next in the algorithm. A *jump* changes the insertion point for exactly one rectangle of the rectangulation. Formally, for a rectangulation $R \in \mathcal{R}_n$, we say that $R' \in \mathcal{R}_n$ differs from R by a *right jump of rectangle r_j by d steps*, denoted $R' = \overrightarrow{J}(R, j, d)$, where $2 \leq j \leq n$ and $d > 0$, if one of the following conditions holds; see Figure 8:

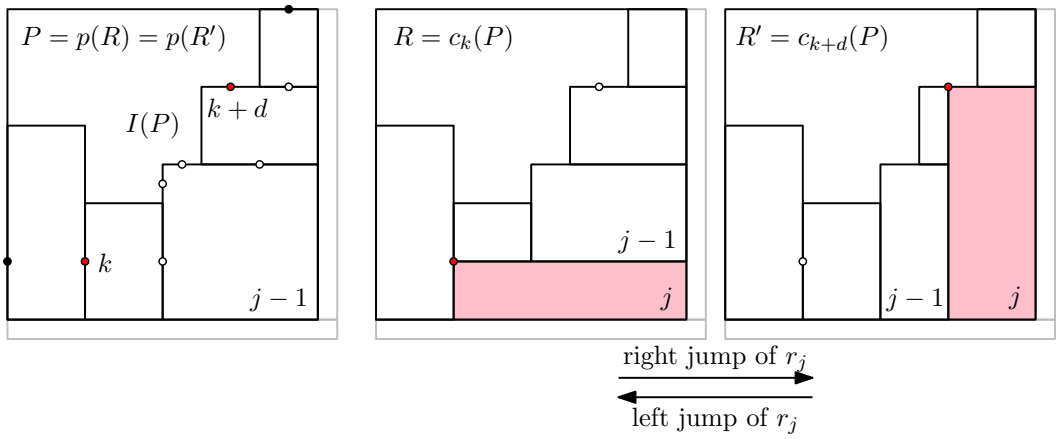


Figure 8 Definition of jumps.

- $j = n$, and $p(R) = p(R') =: P \in \mathcal{R}_{n-1}$, $R = c_k(P)$ and $R' = c_{k+d}(P)$ for some $k > 0$;
- $j < n$, and R and R' are either both bottom-based or both right-based, and $p(R')$ differs from $p(R)$ in a right jump of rectangle r_j by d steps.

In words, the first condition asserts that the first $n-1$ rectangles in R and R' form the same rectangulation $P \in \mathcal{R}_{n-1}$, and R and R' are obtained by insertion from P using the kth and $(k+d)$th insertion point, respectively. The second condition asserts that R and R' agree in the rectangle r_n, which either forms the bottom boundary or the right boundary of those rectangulations, and $p(R')$ differs from $p(R)$ in a right jump with the same parameters.

A right jump as before is called *minimal* w.r.t. to a set of rectangulations $\mathcal{C}_n \subseteq \mathcal{R}_n$, if in the first condition above there is no index ℓ with $k < \ell < k+d$ such that $c_\ell(P) \in \mathcal{C}_n$.

A *(minimal) left jump*, denoted $R' = \overleftarrow{J}(R, j, d)$, is defined analogously by replacing c_{k+d} by c_{k-d} and $k < \ell < k+d$ by $k > \ell > k-d$ in the definitions above. Clearly, if R' differs from R by a right jump of rectangle r_j by d steps, then R differs from R' by a left jump of rectangle r_j by d steps, and vice versa, i.e., we have $R' = \overrightarrow{J}(R, j, d)$ if and only if $R = \overleftarrow{J}(R', j, d)$. We sometimes simply say that R and R' differ in a jump, without specifying the direction. We state the following simple observations for further reference; see Figure 9.

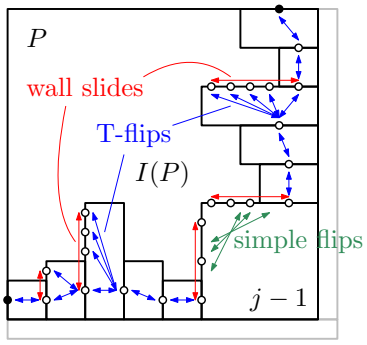


Figure 9 Jumps generalize wall slides, simple flips and T-flips.

▶ **Lemma 3.** *Consider two rectangulations $R, R' \in \mathcal{R}_n$ that differ in a jump of rectangle r_j, define $P := R^{[j-1]} = R'^{[j-1]} \in \mathcal{R}_{j-1}$, and let q_k and q_ℓ be the insertion points in $I(P)$ such that $R^{[j]} = c_k(P)$ and $R'^{[j]} = c_\ell(P)$.*

(a) *If q_k and q_ℓ are consecutive (w.r.t. $I(P)$) on a common wall of P, then R and R' differ in a wall slide.*

(b) *If q_k lies on the last vertical wall and q_ℓ on the first horizontal wall of P (w.r.t. $I(P)$), then R and R' differ in a simple flip.*

(c) *If q_k lies on a vertical wall and q_ℓ is the first insertion point on the next vertical wall of P (w.r.t. $I(P)$), or if q_k lies on a horizontal wall and q_ℓ is the last insertion point on the previous horizontal wall, then R and R' differ in a T-flip.*

3.2 Generating rectangulations by minimal jumps

Consider the following algorithm that attempts to greedily generate a set of rectangulations $\mathcal{C}_n \subseteq \mathcal{R}_n$ using minimal jumps.

Algorithm J$^\square$ (Greedy minimal jumps).

This algorithm attempts to greedily generate a set of rectangulations $\mathcal{C}_n \subseteq \mathcal{R}_n$ using minimal jumps starting from an initial rectangulation $R_0 \in \mathcal{C}_n$.

J1. [Initialize] Visit the initial rectangulation R_0.

J2. [Jump] Generate an unvisited rectangulation from $\mathcal{C}_n$ by performing a minimal jump of the rectangle with maximum index in the most recently visited rectangulation. If no such jump exists, or the jump direction is ambiguous, then terminate. Otherwise visit this rectangulation and repeat J2.

To illustrate how Algorithm J$^\square$ works, consider the set of five rectangulations $\mathcal{C}_4 = \{R_1, \ldots, R_5\} \subseteq \mathcal{R}_4$ shown in Figure 10. If initialized with $R_0 := R_1$, then the algorithm performs a left jump of rectangle 4 by one step (a right jump of rectangle 4 is impossible) to reach R_2, i.e., we have $R_2 = \overleftarrow{J}(R_1, 4, 1)$. In R_2, there are two options, either a right jump of rectangle 4 by one step, leading back to R_1, which has been visited before, or a left jump of rectangle 4 by two steps, leading to R_3, so we visit $R_3 = \overleftarrow{J}(R_2, 4, 2)$. In R_3, the jumps involving rectangle 4 lead to rectangulations that were visited before (R_1 and R_2). Moreover, a jump of rectangle 3 does not lead to a rectangulation in $\mathcal{C}_4$. However, a right jump of rectangle 2 by one step leads to R_4 (a left jump of rectangle 2 is impossible), so we visit $R_4 = \overrightarrow{J}(R_3, 2, 1)$. Finally, in R_4 a right jump of rectangle 4 by two steps leads to $R_5 = \overrightarrow{J}(R_4, 4, 2)$ (a left jump of rectangle 4 is impossible). In this example, Algorithm J$^\square$ successfully visits every rectangulation from $\mathcal{C}_4$ exactly once.

On the other hand, suppose we instead initialize the algorithm with $R_0 := R_3$. The algorithm will then visit $R_2 := \overrightarrow{J}(R_3, 4, 2)$ followed by $R_1 := \overrightarrow{J}(R_2, 4, 1)$, and then terminates without success, as from R_1 no jump leads to an unvisited rectangulation from $\mathcal{C}_4$. Lastly,

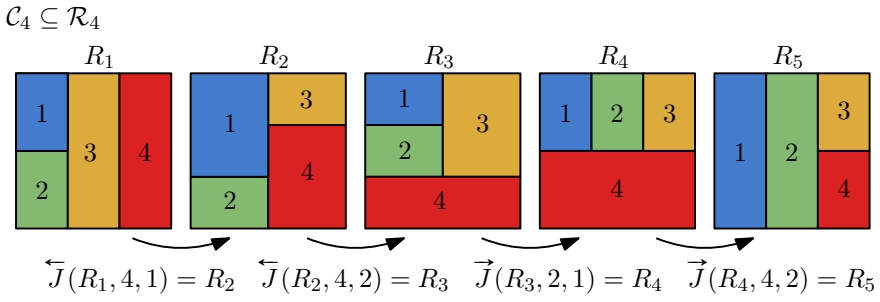


Figure 10 Example execution of Algorithm J$^\square$.

suppose we initialize Algorithm J$^\square$ with $R_0 := R_2$. As before, in R_2, there are two possibilities, either a right jump or a left jump of rectangle 4, both leading to an unvisited rectangulation from $\mathcal{C}_4$. Both are minimal jumps in opposite directions, and as the jump direction is ambiguous, the algorithm terminates immediately without success.

▶ Remark 4. *We do not recommend using Algorithm $J^\square$ in the stated form to generate a set of rectangulations efficiently!* This is because the algorithm requires to maintain the list of all previously visited rectangulations (possibly exponentially many), and to look up this list in each step to check whether a rectangulation obtained by a jump from the current one has been visited before. For us, Algorithm J$^\square$ is merely a tool to define a Gray code ordering of the rectangulations in the given set $\mathcal{C}_n$ in way that is easy to remember (cf. [35]). In fact, in Section 5 we will present a modified algorithm that dispenses with the costly lookup operations, and that computes the very same sequence of rectangulations.

3.3 A guarantee for success

By definition, Algorithm J$^\square$ visits every rectangulation from a given set $\mathcal{C}_n \subseteq \mathcal{R}_n$ at most once, but it may terminate before having visited all. We now provide a sufficient condition guaranteeing that Algorithm J$^\square$ visits every rectangulation from $\mathcal{C}_n$ exactly once.

A set of generic rectangulations $\mathcal{C}_n \subseteq \mathcal{R}_n$ is called *zigzag*, if either $n = 1$ and $\mathcal{C}_1 = \{\square\}$, or if $n \geq 2$ and $\mathcal{C}_{n-1} := \{p(R) \mid R \in \mathcal{C}_n\}$ is zigzag and for every $R \in \mathcal{C}_{n-1}$ we have $c_1(R) \in \mathcal{C}_n$ and $c_{\nu(R)}(R) \in \mathcal{C}_n$. In words, a zigzag set $\mathcal{C}_n$ is closed under repeatedly deleting bottom-right rectangles and replacing them by rectangles inserted either below or to the right of the remaining ones; recall Figure 1. The name "zigzag" does not refer to the shape of a rectangulation and will become clear momentarily. We also say that $\mathcal{C}_n$ is *symmetric*, if reflection at the main diagonal is an involution of $\mathcal{C}_n$, i.e., if $R \in \mathcal{C}_n$, then the rectangulation obtained from R by reflection at the main diagonal is also in $\mathcal{C}_n$. We write  for the rectangulation that consists of n vertically stacked rectangles.

▶ **Theorem 5.** *Given any zigzag set of rectangulations $\mathcal{C}_n$ and initial rectangulation $R_0 =$*  *, Algorithm $J^\square$ visits every rectangulation from $\mathcal{C}_n$ exactly once. Moreover, if $\mathcal{C}_n$ is symmetric, then the ordering of rectangulations generated by Algorithm $J^\square$ is cyclic, i.e., the first and last rectangulation differ in a minimal jump.*

Note that the rectangulation $R_0 =$  is contained in every zigzag set by definition, so this is a valid initialization for Algorithm J$^\square$. We write $J^\square(\mathcal{C}_n)$ for the sequence of rectangulations generated by Algorithm J$^\square$ for a zigzag set $\mathcal{C}_n$ when initialized with $R_0 =$  .

It is easy to see that the number of distinct zigzag sets of generic rectangulations is at least $2^{|\mathcal{R}_n|(1-o(1))} \geq 2^{\Omega(11.56^n)}$ (the latter estimate uses the best known lower bound on $|\mathcal{R}_n|$ from [3]), i.e., at least double-exponential in n. In other words, Algorithm J$^\square$ exhaustively generates a given set of generic rectangulations in a vast number of cases. Moreover, many natural classes of rectangulations are in fact zigzag. In particular, *all* the classes introduced in Section 2.2 and shown in Table 1 satisfy the aforementioned closure property. Moreover, all of these classes are symmetric, so for each of them we obtain cyclic jump orderings.

3.4 Tree of rectangulations

The notion of zigzag sets and the operation of Algorithm J$^\square$ can be interpreted combinatorially in the so-called *tree of rectangulations*, which is an infinite rooted tree, defined recursively as follows; see Figure 11: The root of the tree is a single rectangle $\square \in \mathcal{R}_1$. For any node

$R \in \mathcal{R}_n$, $n \geq 1$, of the tree we consider all insertion points of the rectangulation R, and the set of children of R in the tree is $\{c_i(R) \in \mathcal{R}_{n+1} \mid i = 1, \ldots, \nu(R)\}$. Conversely, the parent of each $R \in \mathcal{R}_n$, $n \geq 2$, is $p(R) \in \mathcal{R}_{n-1}$. In words, insertion leads to the children of a node, and deletion leads to the parent of a node. By Lemma 2, each generic rectangulation appears exactly once in the tree, and the set of nodes in distance n from the root of the tree is precisely the set $\mathcal{R}_{n+1}$ of generic rectangulations with $n+1$ rectangles. We emphasize that this tree is *unordered*, i.e., there is no specified ordering among the children of a node.

By Lemma 1, a node $R \in \mathcal{R}_n$ in the tree has at most $n+1$ children, i.e., we have $|\mathcal{R}_n| \leq n!$. As we see from Figure 11, this inequality is tight up to $n = 4$, but starting from $n = 4$, there are nodes $R \in \mathcal{R}_n$ with strictly less than $n+1$ children, i.e., we have $|\mathcal{R}_5| < 5!$. In fact, it was shown in [3] that $|\mathcal{R}_n| = \mathcal{O}(28.3^n)$.

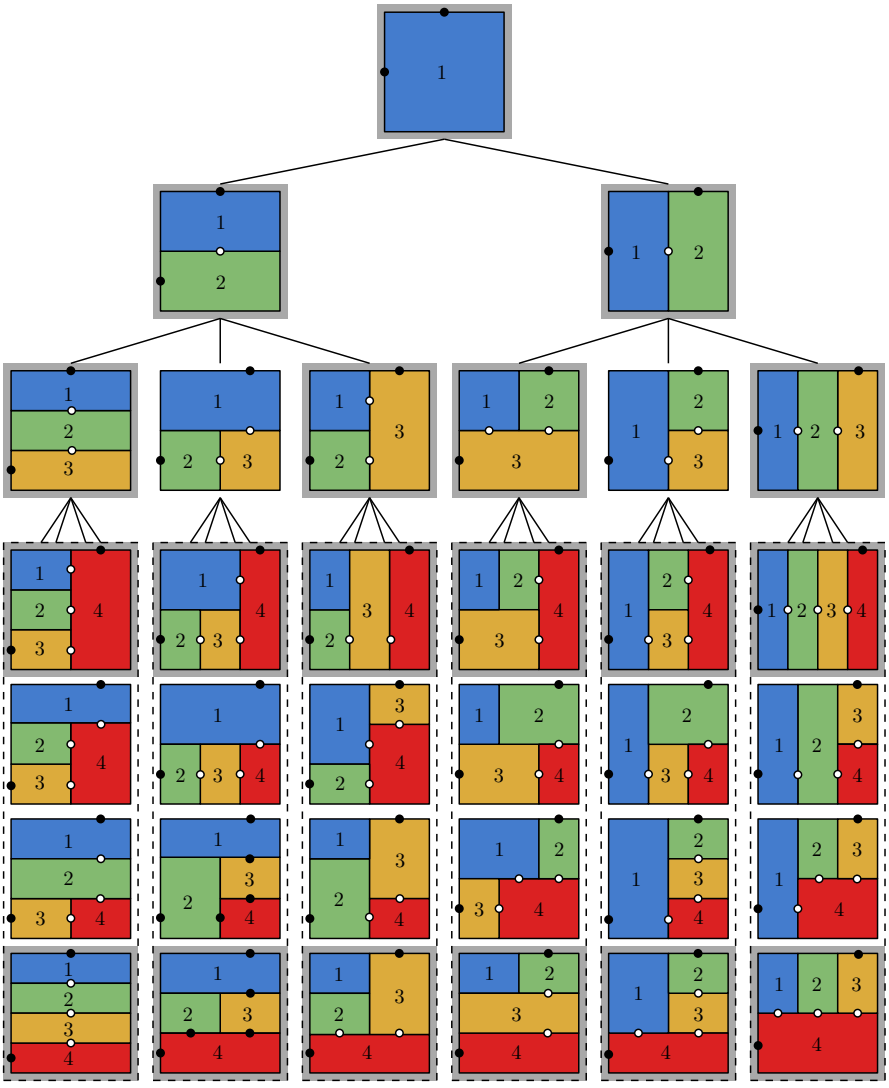


Figure 11 Tree of generic rectangulations up to depth 3 with insertion points highlighted, where first and last insertion point are filled. The rectangulations in the dashed boxes at the bottom level $\mathcal{R}_4$ are stacked on top of each other due to space constraints, but they are children of a common parent node. Bottom- or right-based rectangulations, corresponding to insertion at the first or last insertion point, are marked by gray boxes.

A subset $\mathcal{C}_n \subseteq \mathcal{R}_n$ of nodes in depth $n-1$ of this tree is zigzag, if and only if it arises from the full tree of rectangulations by pruning some subtrees whose roots are neither bottom-based nor right-based rectangulations. In Figure 11, all bottom-based or right-based rectangulations are highlighted by gray boxes, and can therefore not be pruned, while all other nodes can possibly be pruned. If no nodes are pruned, then we have $\mathcal{C}_n = \mathcal{R}_n$, and if all possible nodes are pruned, then $\mathcal{C}_n$ is the set $\mathcal{B}_n$ of 2^{n-1} rectangulations obtained by repeatedly stacking a new rectangle either below or to the right of the previous ones, i.e., $\mathcal{B}_n = \{c_1(R), c_{\nu(R)}(R) \mid R \in \mathcal{B}_{n-1}\}$ for $n \geq 2$ and $\mathcal{B}_1 = \{\square\}$. Moreover, we have $\mathcal{B}_n \subseteq \mathcal{C}_n \subseteq \mathcal{R}_n$ for any zigzag set $\mathcal{C}_n$.

The operation of Algorithm J$^\square$ for a zigzag set $\mathcal{C}_n$ as input can be interpreted as follows: Given the pruned tree corresponding to $\mathcal{C}_n$, we consider the set of nodes on all previous levels of the tree, i.e., the sets $\mathcal{C}_{i-1} := \{p(R) \mid R \in \mathcal{C}_i\}$ for $i = n, n-1, \ldots, 2$, which are all zigzag sets by definition. Moreover, we consider the orderings $J^\square(\mathcal{C}_i)$, $i = 1, \ldots, n$, defined by Algorithm J$^\square$ for each of these sets. These sequences turn the unordered tree corresponding to $\mathcal{C}_n$ into an ordered tree, where the children $c_i(R)$ of each node R from left to right appear alternatingly in increasing order $i = 1, \ldots, \nu(R)$ or in decreasing order $i = \nu(R), \nu(R)-1, \ldots, 1$. Consequently, in the sequence $J^\square(\mathcal{C}_i)$, $i \geq 2$, which forms the left-to-right sequence of all nodes in depth $i-1$ of this ordered tree, the rectangle r_i alternatingly jumps left and right between the first and last insertion point, which motivates the name "zigzag" set; see also the animations provided in [10].

4 Pattern-avoiding rectangulations

We now show that Algorithm J$^\square$ applies to a large number of rectangulation classes that are defined by pattern avoidance, under some very mild conditions; recall Table 1.

A *rectangulation pattern* is a configuration of walls with prescribed directions and incidences. For example, the windmill patterns  and  describe four walls such that when considering the walls in clockwise or counterclockwise order, respectively, the end vertex of one wall lies in the interior of the next wall. We can also think of a pattern as the rectangulation formed by the given walls and incidences. For example, we can think of the windmill patterns as rectangulations with 5 rectangles. We say that a rectangulation R *contains* the pattern P, if R contains a subset of walls with the directions and incidences specified by P. Otherwise we say that R *avoids* P. For any set of rectangulation patterns $\mathcal{P}$ and for any set of rectangulations $\mathcal{C}$, we write $\mathcal{C}(\mathcal{P})$ for the rectangulations from $\mathcal{C}$ that avoid each pattern from $\mathcal{P}$. For example, diagonal rectangulations are given by $\mathcal{D}_n = \mathcal{R}_n(\{$  ,  $\})$.

We say that a rectangulation pattern P is *tame*, if for any rectangulation R that avoids P, we also have that $c_1(R)$ and $c_{\nu(R)}(R)$ avoid P. In words, inserting a new rectangle below R or to the right of R must not create the pattern P. These definitions yield the next lemma.

▶ **Lemma 6.** *If a rectangulation pattern is neither bottom-based nor right-based, then it is tame. In particular, each of the patterns*  *is tame.*

The following powerful theorem allows to obtain many new zigzag sets of rectangulations from a given zigzag set $\mathcal{C}_n \subseteq \mathcal{R}_n$ by forbidding one or more tame patterns. All of these zigzag sets can then be generated by our Algorithm J$^\square$.

▶ **Theorem 7.** *Let $\mathcal{C}_n \subseteq \mathcal{R}_n$ be a zigzag set of rectangulations, and let $\mathcal{P}$ be a set of tame rectangulation patterns. Then $\mathcal{C}_n(\mathcal{P})$ is a zigzag set of rectangulations. Moreover, if $\mathcal{P}$ is symmetric, then $\mathcal{C}_n(\mathcal{P})$ is symmetric.*

Recall that $\mathcal{P}$ is symmetric if for each pattern $P \in \mathcal{P}$, we have that the pattern obtained from P by reflection at the main diagonal is also in $\mathcal{P}$. The significance of the second part of the theorem is that if $\mathcal{C}_n(\mathcal{P})$ is symmetric, then the ordering of rectangulations of $\mathcal{C}_n(\mathcal{P})$ generated by Algorithm $\mathrm{J}^{\square}$ is cyclic by Theorem 5. See [23] for a proof of Theorem 7.

5 Efficient computation

Recall from Remark 4 that Algorithm $\mathrm{J}^{\square}$ in its stated form is unsuitable for efficient implementation. We now discuss how to make the algorithm efficient, so as to achieve the time bounds claimed in Table 1 for several interesting classes of rectangulations.

5.1 Memoryless algorithm

Consider Algorithm $\mathrm{M}^{\square}$ below, which takes as input a zigzag set of rectangulations $\mathcal{C}_n \subseteq \mathcal{R}_n$ and generates them exhaustively by minimal jumps in the same order as Algorithm $\mathrm{J}^{\square}$, i.e., in the order $J^{\square}(\mathcal{C}_n)$. After initialization in line M1, the algorithm loops over lines M2–M5, visiting the current rectangulation R at the beginning of each iteration (line M2), until it terminates (line M3). The key idea is to track explicitly which rectangle jumps in each step, and the direction of the jump. With this information, the jump is determined by the condition that it must be minimal w.r.t. $\mathcal{C}_n$, i.e., starting from the current insertion point of the given rectangle, we choose the first insertion point (w.r.t. their linear ordering) for that rectangle in the given direction that creates the next rectangulation from $\mathcal{C}_n$.

Algorithm $\mathrm{M}^{\square}$ (Memoryless minimal jumps).

This algorithm generates all rectangulations of a zigzag set $\mathcal{C}_n \subseteq \mathcal{R}_n$ by minimal jumps in the same order as Algorithm $\mathrm{J}^{\square}$. It maintains the current rectangulation in the variable R, and auxiliary arrays $o = (o_1, \ldots, o_n)$ and $s = (s_1, \ldots, s_n)$.

M1. [Initialize] Set $R \leftarrow$  , and $o_j \leftarrow \triangleleft$, $s_j \leftarrow j$ for $j = 1, \ldots, n$.

M2. [Visit] Visit the current rectangulation R.

M3. [Select rectangle] Set $j \leftarrow s_n$, and terminate if $j = 1$.

M4. [Jump rectangle] In the current rectangulation R, perform a jump of rectangle r_j that is minimal w.r.t. $\mathcal{C}_n$, where the jump direction is left if $o_j = \triangleleft$ and right if $o_j = \triangleright$.

M5. [Update o and s] Set $s_n \leftarrow n$. If $o_j = \triangleleft$ and $R^{[j]}$ is bottom-based set $o_j \leftarrow \triangleright$, or if $o_j = \triangleright$ and $R^{[j]}$ is right-based set $o_j \leftarrow \triangleleft$, and in both cases set $s_j \leftarrow s_{j-1}$ and $s_{j-1} \leftarrow j - 1$. Go back to M2.

Specifically, the jump directions are maintained by an array $o = (o_1, \ldots, o_n)$, where $o_j = \triangleleft$ means that rectangle r_j performs a left jump in the next step, and $o_j = \triangleright$ means that rectangle r_j performs a right jump in the next step (line M4). All sub-rectangulations of the initial rectangulation  are right-based, so the initial jump directions are $o_j = \triangleleft$ for $j = 1, \ldots, n$ (line M1). Whenever rectangle r_j jumps left and reaches the first insertion point, which means that $R^{[j]}$ is bottom-based, or if it jumps right and reaches the last insertion point, which means that $R^{[j]}$ is right-based, then the jump direction o_j is reversed (line M5).

The array $s = (s_1, \ldots, s_n)$ is used to determine which rectangle jumps in each step. Specifically, the last entry s_n determines the rectangle that jumps in the current iteration (line M3). This array simulates a stack in a loopless fashion, following an idea first used by Bitner, Ehrlich, and Reingold [7]. The stack is initialized by $(s_1, \ldots, s_n) = (1, 2, \ldots, n)$ (line M1), with s_n being the value on the top of the stack. The stack is popped (by the

instruction $s_j \leftarrow s_{j-1}$ in line M5) when rectangle r_j reaches its first or last insertion point in this step, meaning that this rectangle is not eligible to jump in the next step, but becomes eligible again after the next step, which is achieved by pushing the value j on the stack again (by the instructions $s_n \leftarrow n$ and $s_{j-1} \leftarrow j-1$ in line M5). See Table 2 for an example.

▶ **Theorem 8.** *For any zigzag set of rectangulations $\mathcal{C}_n \subseteq \mathcal{R}_n$, Algorithm $M^{\square}$ visits every rectangulation from $\mathcal{C}_n$ exactly once, in the order $J^{\square}(\mathcal{C}_n)$ defined by Algorithm $J^{\square}$.*

To make meaningful statements about the running time of Algorithm M$^{\square}$, we need to specify the data structures used to represent the current rectangulation R, and the operations on this data structure to perform the operations in lines M4 and M5. Most importantly, we need to develop oracles which efficiently compute the next minimal jump w.r.t. $\mathcal{C}_n$ for some interesting zigzag sets $\mathcal{C}_n$. One should think of $\mathcal{C}_n$ here as a class of rectangulations specified by some properties or forbidden patterns, such as "diagonal guillotine rectangulations", and not as large precomputed set of rectangulations. All of these details can be found in [23], and they are part of our C++ implementation provided in [10].

Table 2 Execution of Algorithm M$^{\square}$ for the set $\mathcal{C}_4 = \mathcal{D}_4$ of diagonal rectangulations with 4 rectangles. Empty entries in the o and s column are unchanged compared to the previous row.

	$J^{\square}(\mathcal{C}_4)$	jump	$o_1o_2o_3o_4$	$s_1s_2s_3s_4$		$J^{\square}(\mathcal{C}_4)$	jump	$o_1o_2o_3o_4$	$s_1s_2s_3s_4$
1		$\overleftarrow{J}(R,4,1)$	◁ ◁ ◁ ◁	1 2 3 4	12		$\overrightarrow{J}(R,4,1)$	▷	1 1 4
2		$\overleftarrow{J}(R,4,1)$		4	13		$\overrightarrow{J}(R,4,1)$		4
3		$\overleftarrow{J}(R,4,1)$		4	14		$\overrightarrow{J}(R,4,1)$		4
4		$\overleftarrow{J}(R,3,1)$	▷	3 3	15		$\overrightarrow{J}(R,3,1)$	◁	3 3
5		$\overrightarrow{J}(R,4,1)$		4	16		$\overleftarrow{J}(R,4,1)$		4
6		$\overrightarrow{J}(R,4,1)$		4	17		$\overleftarrow{J}(R,4,1)$		4
7		$\overrightarrow{J}(R,4,1)$		4	18		$\overleftarrow{J}(R,4,1)$		4
8		$\overleftarrow{J}(R,3,1)$	◁	3 3	19		$\overrightarrow{J}(R,3,1)$	▷	3 3
9		$\overleftarrow{J}(R,4,1)$	▷	2 2 4	20		$\overrightarrow{J}(R,4,1)$	◁	2 1 4
10		$\overleftarrow{J}(R,4,2)$		4	21		$\overrightarrow{J}(R,4,2)$		4
11		$\overleftarrow{J}(R,2,1)$	▷	3 2	22			◁	3 1

References

1 E. Ackerman, G. Barequet, and R. Y. Pinter. A bijection between permutations and floorplans, and its applications. *Discrete Appl. Math.*, 154(12):1674–1684, 2006. doi:10.1016/j.dam.2006.03.018.

2 E. Ackerman, G. Barequet, and R. Y. Pinter. On the number of rectangulations of a planar point set. *J. Combin. Theory Ser. A*, 113(6):1072–1091, 2006. doi:10.1016/j.jcta.2005.10.003.

3 K. Amano, S. Nakano, and K. Yamanaka. On the number of rectangular drawings: Exact counting and lower and upper bounds, 2007. IPSJ SIG Technical Report 2007-AL-115 (5).

4 A. Asinowski, G. Barequet, M. Bousquet-Mélou, T. Mansour, and R. Y. Pinter. Orders induced by segments in floorplans and (2-14-3, 3-41-2)-avoiding permutations. *Electron. J. Combin.*, 20(2):Paper 35, 43, 2013.

5 A. Asinowski and T. Mansour. Separable d-permutations and guillotine partitions. *Ann. Comb.*, 14(1):17–43, 2010. doi:10.1007/s00026-010-0043-8.

6 D. Avis and K. Fukuda. Reverse search for enumeration. *Discrete Appl. Math.*, 65(1-3):21–46, 1996. First International Colloquium on Graphs and Optimization (GOI), 1992 (Grimentz). doi:10.1016/0166-218X(95)00026-N.

7 J. R. Bitner, G. Ehrlich, and E. M. Reingold. Efficient generation of the binary reflected Gray code and its applications. *Comm. ACM*, 19(9):517–521, 1976. doi:10.1145/360336.360343.

8 J. Cardinal, V. Sacristán, and R. I. Silveira. A note on flips in diagonal rectangulations. *Discrete Math. Theor. Comput. Sci.*, 20(2):Paper No. 14, 22, 2018.

9 J. Conant and T. Michaels. On the number of tilings of a square by rectangles. *Ann. Comb.*, 18(1):21–34, 2014. doi:10.1007/s00026-013-0209-2.

10 The Combinatorial Object Server: http://www.combos.org/rect.

11 G. Ehrlich. Loopless algorithms for generating permutations, combinations, and other combinatorial configurations. *J. Assoc. Comput. Mach.*, 20:500–513, 1973. doi:10.1145/321765.321781.

12 D. Eppstein, E. Mumford, B. Speckmann, and K. Verbeek. Area-universal and constrained rectangular layouts. *SIAM J. Comput.*, 41(3):537–564, 2012. doi:10.1137/110834032.

13 R. Fujimaki, Y. Inoue, and T. Takahashi. An asymptotic estimate of the numbers of rectangular drawings or floorplans. In *2009 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 856–859, 2009. doi:10.1109/ISCAS.2009.5117891.

14 E. Hartung, H. P. Hoang, T. Mütze, and A. Williams. Combinatorial generation via permutation languages. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 1214–1225. SIAM, 2020. doi:10.1137/1.9781611975994.74.

15 E. Hartung, H. P. Hoang, T. Mütze, and A. Williams. Combinatorial generation via permutation languages. I. Fundamentals, 2020. To appear in *Trans. Amer. Math. Soc.*; preprint available at arXiv:1906.06069.

16 B. D. He. A simple optimal binary representation of mosaic floorplans and Baxter permutations. *Theoret. Comput. Sci.*, 532:40–50, 2014. doi:10.1016/j.tcs.2013.05.007.

17 H. P. Hoang and T. Mütze. Combinatorial generation via permutation languages. II. Lattice congruences, 2020. To appear in *Israel J. Math.*; preprint available at arXiv:1911.12078.

18 X. Hong, G. Huang, Y. Cai, J. Gu, S. Dong, C.-K. Cheng, and J. Gu. Corner block list: An effective and efficient topological representation of non-slicing floorplan. In E. Sentovich, editor, *Proceedings of the 2000 IEEE/ACM International Conference on Computer-Aided Design, 2000, San Jose, California, USA, November 5-9, 2000*, pages 8–12. IEEE Computer Society, 2000. doi:10.1109/ICCAD.2000.896442.

19 Y. Inoue, T. Takahashi, and R. Fujimaki. Counting rectangular drawings or floorplans in polynomial time. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, 92-A(4):1115–1120, 2009. doi:10.1587/transfun.E92.A.1115.

20 D. E. Knuth. *The Art of Computer Programming. Vol. 4A. Combinatorial algorithms. Part 1.* Addison-Wesley, Upper Saddle River, NJ, 2011.

21 S. Law and N. Reading. The Hopf algebra of diagonal rectangulations. *J. Combin. Theory Ser. A*, 119(3):788–824, 2012. doi:10.1016/j.jcta.2011.09.006.

22 E. Meehan. The Hopf algebra of generic rectangulations, 2019. arXiv:1903.09874.

23 A. Merino and T. Mütze. Combinatorial generation via permutation languages. III. Rectangulations, 2021. Full preprint version of the present article. arXiv:2103.09333.

24 W. J. Mitchell, J. P. Steadman, and R. S. Liggett. Synthesis and optimization of small rectangular floor plans. *Environment and Planning B: Planning and Design*, 3(1):37–70, 1976. doi:10.1068/b030037.

25 S. Nakano. Enumerating floorplans with n rooms. In *Algorithms and computation (Christchurch, 2001)*, volume 2223 of *Lecture Notes in Comput. Sci.*, pages 107–115. Springer, Berlin, 2001. doi:10.1007/3-540-45678-3_10.

26 OEIS Foundation Inc. The on-line encyclopedia of integer sequences, 2020. URL: http://oeis.org.

27 R. H. J. M. Otten. Automatic floorplan design. In J. S. Crabbe, C. E. Radke, and H. Ofek, editors, *Proceedings of the 19th Design Automation Conference, DAC '82, Las Vegas, Nevada, USA, June 14-16, 1982*, pages 261–267. ACM/IEEE, 1982. doi:10.1145/800263.809216.

28 V. Pilaud and F. Santos. Quotientopes. *Bull. Lond. Math. Soc.*, 51(3):406–420, 2019. doi:10.1112/blms.12231.

29 N. Reading. Generic rectangulations. *European J. Combin.*, 33(4):610–623, 2012. doi:10.1016/j.ejc.2011.11.004.

30 F. Ruskey. Combinatorial Gray code. In M.-Y. Kao, editor, *Encyclopedia of Algorithms*, pages 342–347. Springer, 2016.

31 C. Savage. A survey of combinatorial Gray codes. *SIAM Rev.*, 39(4):605–629, 1997. doi:10.1137/S0036144595295272.

32 Z. C. Shen and C. C. N. Chu. Bounds on the number of slicing, mosaic, and general floorplans. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 22(10):1354–1361, 2003. doi:10.1109/TCAD.2003.818136.

33 M. Takagi and S. Nakano. Listing all rectangular drawings with certain properties. *Systems and Computers in Japan*, 35(4):1–8, 2004. doi:10.1002/scj.10563.

34 M. van Kreveld and B. Speckmann. On rectangular cartograms. *Comput. Geom.*, 37(3):175–187, 2007. doi:10.1016/j.comgeo.2006.06.002.

35 A. Williams. The greedy Gray code algorithm. In *Algorithms and Data Structures - 13th International Symposium, WADS 2013, London, ON, Canada, August 12-14, 2013. Proceedings*, pages 525–536, 2013. doi:10.1007/978-3-642-40104-6_46.

36 K. Yamanaka, M. S. Rahman, and S. Nakano. Enumerating floorplans with columns. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, 101-A(9):1392–1397, 2018. doi:10.1587/transfun.E101.A.1392.

37 B. Yao, H. Chen, C.-K. Cheng, and R. L. Graham. Floorplan representations: Complexity and connections. *ACM Trans. Design Autom. Electr. Syst.*, 8(1):55–80, 2003. doi:10.1145/606603.606607.

38 S. Yoshii, D. Chigira, K. Yamanaka, and S. Nakano. Constant time generation of rectangular drawings with exactly n faces. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, 89-A(9):2445–2450, 2006. doi:10.1093/ietfec/e89-a.9.2445.

Polygon-Universal Graphs

Tim Ophelders
Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands

Ignaz Rutter
Department of Computer Science and Mathematics, Universität Passau, Germany

Bettina Speckmann
Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands

Kevin Verbeek
Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands

Abstract

We study a fundamental question from graph drawing: given a pair (G, C) of a graph G and a cycle C in G together with a simple polygon P, is there a straight-line drawing of G inside P which maps C to P? We say that such a drawing of (G, C) *respects* P. We fully characterize those instances (G, C) which are *polygon-universal*, that is, they have a drawing that respects P for any simple (not necessarily convex) polygon P. Specifically, we identify two necessary conditions for an instance to be polygon-universal. Both conditions are based purely on graph and cycle distances and are easy to check. We show that these two conditions are also sufficient. Furthermore, if an instance (G, C) is planar, that is, if there exists a planar drawing of G with C on the outer face, we show that the same conditions guarantee for every simple polygon P the existence of a planar drawing of (G, C) that respects P. If (G, C) is polygon-universal, then our proofs directly imply a linear-time algorithm to construct a drawing that respects a given polygon P.

2012 ACM Subject Classification Human-centered computing → Graph drawings

Keywords and phrases Graph drawing, partial drawing extension, simple polygon

Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.55

Related Version *Full Version*: `https://arxiv.org/abs/2103.06916`

Funding Research on the topic of this paper was initiated at the 2nd Workshop on Applied Geometric Algorithms in Vierhouten, NL, supported by the Dutch Research Council (NWO); 639.023.208.
Ignaz Rutter: Partially supported by the German Science Foundation (DFG); Ru 1903/3-1.
Bettina Speckmann: Partially supported by the Dutch Research Council (NWO); 639.023.208.

Acknowledgements Ignaz Rutter would like to thank Michael Hoffmann and Vincent Kusters for discussions on conjectures related to this paper.

1 Introduction

Graphs are a convenient way to express relations between entities. To visualize these relations, the corresponding graph needs to be *drawn*, most commonly in the plane and with straight edges. Naturally there are a multitude of different optimization criteria and drawing restrictions that attempt to capture various perceptual requirements or real-world conditions. In this paper we focus on drawings which are constrained to the interiors of simple polygons.

The *polygon-extension problem* asks, whether a given graph admits a (planar) drawing where the outer face is fixed to a given simple polygon P; see Fig. 1 for examples. Our main focus is the *polygon-universality problem*, which asks whether a given plane graph admits a polygon-extension for every choice of fixing the outer face to a simple polygon. As is often the case with geometric problems, a natural complexity class for the polygon-extension problem is $\exists\mathbb{R}$, the class of problems that can be encoded in polynomial time as an

existentially quantified formula of real variables (rather than Boolean variables as for SAT), which was introduced by Schaefer and Štefankovič [12]. The natural complexity class for the polygon-universality problem is $\forall\exists\mathbb{R}$, the universal existential theory of the reals, which has been recently defined by Dobbins et al. [5]. It is known that NP $\subseteq \exists\mathbb{R} \subseteq \forall\exists\mathbb{R} \subseteq$ PSPACE [3].

Tutte [13] proved that there is a straight-line planar drawing of a planar graph G inside an arbitrary convex polygon P if one fixes the outer face of (an arbitrary planar embedding of) G to P. This result has been generalized to allow polygons P that are non-strictly convex [4, 6] or even star-shaped polygons [7]. These results have applications in *partial drawing extension* problems. Here, in addition to an input graph G, we are given a subgraph $H \subseteq G$ together with a fixed drawing Γ of H. The question is whether one can extend the given drawing Γ to a planar straight-line drawing of the whole graph G by drawing the vertices and edges of $G - H$ inside the faces of H. If the embedding of G is fixed, the results by Tutte and others allow to reduce the problem by removing vertices of G that are contained in convex or star-shaped faces of Γ. Such reduction rules have lead to efficient testing algorithms for special cases, for example, when the drawing of H is convex [10].

Recently, Lubiw et al. [9] showed that it is $\exists\mathbb{R}$-complete to decide for a given planar graph that is partially fixed to a non-crossing polygon *with holes*, whether the partial drawing can be extended to a planar straight-line drawing that does not intersect the outside of the polygon. That is, the planar polygon extension problem is $\exists\mathbb{R}$-complete for polygons with holes. They leave the case of simple polygons open.

If we do not insist on straight-line drawings, then other questions arise. Angelini et al. [2] give an $O(mn)$-time algorithm for testing whether an n-vertex outer-planar graph admits a planar one-bend drawing whose outer face is fixed to a simple polygon on m vertices. Mchedlidze and Urhausen [11] link the number of bends per edge that are necessary for extending a drawing to a convexity measure for the faces of the partial drawing. Angelini et al. [1] present a linear time algorithm to decide if a partially fixed drawing has a planar drawing using Jordan arcs, while Jelínek et al. [8] characterize the solvable instances by forbidden substructures.

Quite recently, Dobbins et al. [5] considered the problem of area-universality for graphs with partial drawings. Let G be a planar graph with a fixed embedding, including the outer face. An assignment of areas to faces of G is *realizable* if G admits a straight-line drawing such that each face has the assigned area. A graph is *area-universal* if every area assignment is realizable. Dobbins et al. prove that it is $\exists\mathbb{R}$-complete to decide whether an area assignment is realizable for a planar triangulation, which has been partially drawn, and testing area-universality is $\forall\exists\mathbb{R}$-complete if the planarity condition is dropped (but still parts of the drawing are fixed). They conjecture that the same area-universality problems without a partially fixed drawing are $\exists\mathbb{R}$- and $\forall\exists\mathbb{R}$-complete in general.

Notation. Let $G = (V, E)$ be a graph with n vertices. A *drawing* $\mathcal{D}$ of G is a map from each $v \in V$ to points in the plane and from each edge $e \in E$ to a Jordan arc connecting its endpoints. A *straight-line* drawing maps each edge to a straight line segment. A drawing is *planar*, if no two edges intersect, except at common endpoints. A graph G is planar if it has a planar drawing. Let $C = [c_1, \ldots, c_t]$ with $c_i \in V$ be a simple cycle in G. An *instance* (G, C) is planar if G has a planar drawing with C as the outer face. Let P be a simple polygon with t vertices $[p_1, \ldots, p_t]$ with $p_i \in \mathbb{R}^2$. A drawing $\mathcal{D}$ of (G, C) *respects* P if it is a map $\mathcal{D}\colon V \to P$ from vertices to points in P such that $\mathcal{D}(c_i) = p_i$ and for each edge $\{u, v\} \in E$, the line segment between $\mathcal{D}(u)$ and $\mathcal{D}(v)$ lies in P (see Figure 1). That is, $\mathcal{D}$ is a straight-line

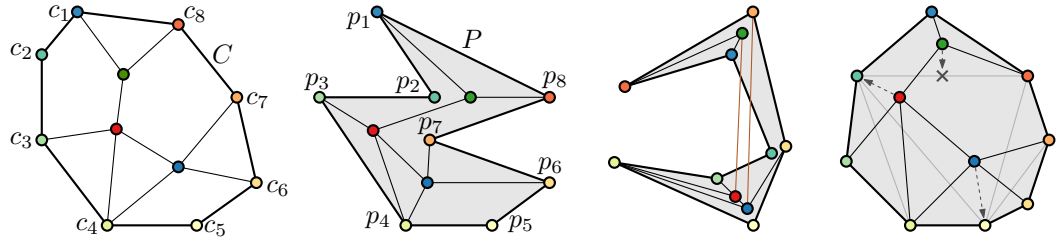


Figure 1 Left: an instance (G, C). Center left: a drawing of (G, C) that respects a polygon P (shaded in grey). Center right: there is no drawing of (G, C) that respects this polygon. Right: A triangulated convex polygon with a drawing that is not triangulation-respecting; moving the vertices along the dashed arrows results in a triangulation-respecting drawing.

drawing of G inside P that fixes the vertices of C to the corresponding vertices of P. An instance (G, C) is *(planar) polygon-universal* if it admits a (planar) straight-line drawing that respects every simple (not necessarily convex) polygon P on t vertices.

Our algorithms use a triangulation $\mathcal{T}$ of P to construct a drawing or prove the non-universality. We say that a drawing of (G, C) *respects* $\mathcal{T}$ if no edge of G properly crosses an edge of $\mathcal{T}$. Although every triangulation-respecting drawing is also a drawing, the converse is not true. In fact, there are graphs G, cycles C, and polygons P that have a drawing, but no triangulation of P exists that allows a triangulation-respecting drawing (see Figure 2).

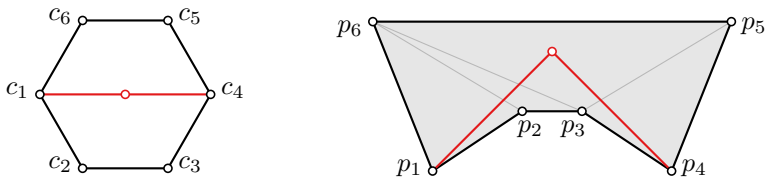


Figure 2 A graph G and a polygon for which there exists a drawing of G, but no triangulation with a triangulation-respecting drawing.

Results and organization. In Section 2 we identify two necessary conditions for an instance (G, C) to be *polygon-universal*. These conditions are purely based on graph and cycle distances and hence easy to check. To show that these two conditions are also sufficient, we use triangulation-respecting drawings: if there is a triangulation $\mathcal{T}$ of a simple polygon P such that (G, C) does not admit a triangulation-respecting drawing for $\mathcal{T}$, then we can argue that G contains one of two forbidden substructures, violating the necessary conditions (see Section 5). These substructures certify that (G, C) is not polygon-universal.

To arrive at this conclusion, in Section 3 we first present an algorithm that tests in linear time for a given instance (G, C), a polygon P, and triangulation $\mathcal{T}$ of P, whether there exists a triangulation-respecting drawing of G for $\mathcal{T}$ inside P. If so, we can construct the drawing in linear time. Then, in Section 4, we consider planar instances (G, C) and show that the same algorithm can decide in linear time whether there is a triangulation-respecting drawing that is planar after infinitesimal perturbation. An analysis of this algorithm shows that a planar instance (G, C) is planar polygon-universal if and only if it is polygon-universal.

2 Necessary conditions for polygon-universality

We present two necessary conditions for an instance (G, C) to be polygon-universal. Intuitively, both conditions capture that there need to be "enough" vertices in G between cycle vertices for the drawing not to become "too tight". The *Pair Condition* captures this for any two vertices on the cycle C. The *Triple Condition* is a bit more involved: even if the Pair Condition is satisfied for any pair of vertices on the cycle, there can still be triples of vertices which together "pull too much" on the graph. Specifically, for an instance (G, C) of a graph G and a cycle $C \subset G$ with t vertices, we denote by $d_G \colon V \times V \to \mathbb{N}$ the graph distance in G and by $d_C \colon V(C) \times V(C) \to \mathbb{N}$ the distance (number of edges) along the cycle C. The following conditions are necessary for (G, C) to be polygon-universal for all simple polygons P:

Pair For all i and j, we have $d_C(c_i, c_j) \leq d_G(c_i, c_j)$ (and hence $d_C(c_i, c_j) = d_G(c_i, c_j)$).

Triple For all vertices $v \in V$ and distinct i, j, k with $d_C(c_i, c_j) + d_C(c_j, c_k) + d_C(c_i, c_k) \geq t$ (and hence $= t$), we have $d_G(c_i, v) + d_G(c_j, v) + d_G(c_k, v) > t/2$.

To establish that these two conditions are necessary, we use the *link distance* between two points inside certain simple polygons P. Specifically, the link distance of two points q_1 and q_2 with respect to a simple polygon P is the minimum number of segments for a polyline π that lies inside P and connects q_1 and q_2. If the Pair Condition is violated for two cycle vertices c_i and c_j, we can construct a *Pair Spiral* polygon P (see Figure 3 (left)) such that the link distance between p_i and p_j (the vertices of P to which c_i and c_j are mapped) exceeds $d_G(c_i, c_j)$. Clearly there is no drawing (G, C) that respects P.

If the first condition holds, but the second condition is violated by a vertex v, consider the shortest paths via v that connect c_i, c_j, c_k to each other. By assumption the total length of these three paths is $2d_G(c_i, v) + 2d_G(c_j, v) + 2d_G(c_k, v) \leq t$, while the total length of the paths connecting c_i, c_j, c_k to each other along C is $d_C(c_i, c_j) + d_C(c_j, c_k) + d_C(c_i, c_k) = t$. Since the pair condition holds, the paths via v are not shorter than the paths along C, and therefore the paths via v must be shortest paths connecting the pairs. That is, $d_C(c_i, c_j) = d_G(c_i, v) + d_G(c_j, v)$, $d_C(c_j, c_k) = d_G(c_j, v) + d_G(c_k, v)$ and $d_C(c_i, c_k) = d_G(c_i, v) + d_G(c_k, v)$. In that case, we can construct a *Triple Spiral* polygon P (see Figure 3 (right)) such that there is no point that lies within link-distance $d_G(c_i, v)$ from c_i, link-distance $d_G(c_j, v)$ from c_j, and link-distance $d_G(c_k, v)$ from c_k simultaneously. Hence, there exists no drawing of the aforementioned shortest paths via v that respects P.

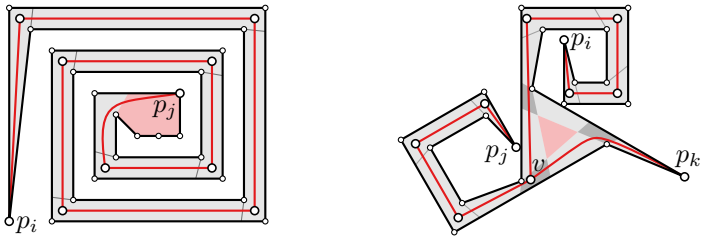


Figure 3 Left: Pair Spiral. Points with link-distance greater than $d_G(c_i, c_j)$ from p_i shaded red. Right: Triple Spiral. Points of link-distance $\leq d_G(c_x, v)$ from p_x for one $x \in \{i, j, k\}$ in light gray; for two $x \in \{i, j, k\}$ in dark gray; there is no point q in P with $d_G(c_x, q) \leq d_G(c_x, v)$ for all $x \in \{i, j, k\}$.

3 Triangulation-respecting drawings

In this section we are given the following input: an instance (G, C) consisting of a graph G with n vertices and a cycle C with t vertices, and a simple polygon P with t vertices together with an arbitrary triangulation $\mathcal{T}$ of P. We study the following question: is there a drawing of (G, C) that respects both P and $\mathcal{T}$?

We describe a dynamic programming algorithm which can answer this question in linear time. The basic idea is as follows: every edge of $\mathcal{T}$ defines a *pocket* of P. We recursively *sketch* a drawing of G within each pocket. Such a sketch assigns an approximate location, such as an edge or a triangle, to each vertex. Ultimately we combine the location constraints on vertex positions posed by the sketches and decide if they can be satisfied.

We root (the dual tree of) $\mathcal{T}$ at an arbitrary triangle T_{root}. Each edge e of $\mathcal{T}$ partitions P into two regions, one of which contains T_{root}. Let Q be the region not containing T_{root}. We say that Q is a *pocket* with the *lid* $e = e_Q$, and we denote the unique triangle outside Q adjacent to e_Q by T_Q^+. Since a pocket is uniquely defined by its lid, we will for an edge e also write Q_e to denote the pocket with lid e. We say that a pocket is *trivial* if its lid lies on the boundary of P; in such case the pocket consists of only that edge. If Q is a non-trivial pocket, then we denote the unique triangle inside Q adjacent to e_Q by T_Q (see Figure 4).

For ease of explanation we consider all indices on C and P modulo t, that is, we identify c_i and c_{i+t} as well as p_i and p_{i+t}. Moreover, when talking about a non-trivial pocket Q with lid (p_i, p_j), whose third vertex of T_Q is p_k, we will assume that $i \leq k \leq j$ (otherwise simply shift the indices cyclically). We first define triangulation-respecting drawings for pockets:

▶ **Definition 1.** *A* triangulation-respecting drawing *for a pocket Q with lid $e_Q = (p_i, p_j)$ is an assignment of the vertices of G to locations inside the polygon P, such that*

1. *Any vertex c_ℓ with $i \leq \ell \leq j$ is assigned to the polygon vertex p_ℓ.*
2. *For any edge (u, v) of G, u and v lie on a common triangle (or edges or vertices thereof).*

We consider the triangles of the triangulation as closed, so that distinct triangles may share a segment (namely an edge of the triangulation) or a point (namely a vertex of P). We define a triangulation-respecting drawing for the entire triangulation analogously, requiring that c_ℓ is assigned to p_ℓ for all ℓ.

A *sketch* is an assignment of the vertices of G to simplices (vertices, edges, or triangles) of the triangulation with the property that, if we draw each vertex anywhere on its assigned simplex, then the result is a triangulation-respecting drawing. We hence interpret a simplex as a closed region of the plane in the remainder of this paper.

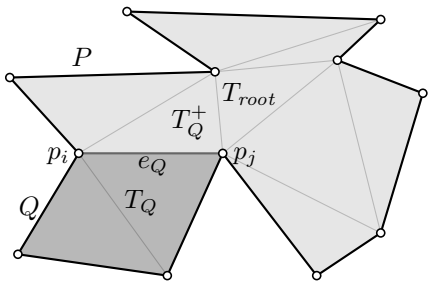


Figure 4 A triangulation with labels for the pocket Q (shaded dark) and triangles T_Q and T_Q^+ incident to edge e_Q of the triangulation.

▶ **Definition 2.** *A* sketch *of the triangulation is a function* Γ *that assigns vertices of* G *to simplices of* $\mathcal{T}$*, such that* (i) *for any vertex* c_i *of the cycle,* $\Gamma(c_i) = p_i$*, and* (ii) *for any two adjacent vertices* u *and* v*, there exists a triangle of* $\mathcal{T}$ *that contains both* $\Gamma(u)$ *and* $\Gamma(v)$*. A sketch of a pocket is defined similarly, except that vertices* p_i *of the polygon that lie outside the pocket do not need* c_i *assigned to them.*

We show that a sketch exists (for a pocket or a triangulation) if and only if there is a triangulation-respecting drawing (for that pocket or triangulation). If a pocket admits a sketch, we call a pocket *sketchable*. If a particular pocket is sketchable, then so are all of its subpockets, since any sketch for a pocket is also a sketch for any of its subpockets.

We present an algorithm that for any sketchable pocket constructs a sketch, and for any other pocket reports that it is not sketchable. This algorithm recursively constructs particularly well-behaved sketches for child pockets, and combines these sketches into a new well-behaved sketch. To obtain a sketch for $\mathcal{T}$, we combine the three well-behaved sketches for the three pockets of the root triangle T_{root} – assuming that all three pockets are sketchable.

Well-behaved sketches. We restrict our attention to *local* sketches for a pocket Q, which assign vertices either to simplices in Q or to the triangle T_Q^+ just outside Q, and *interior* local sketches, which assign vertices to simplices in Q only.

▶ **Lemma 3.** *If there is a sketch for pocket* Q*, then there is a local sketch for* Q*.*

Generally, it is advantageous for a sketch to place its vertices as far "to the outside" as possible, to generate maximum flexibility when combining sketches. Hence, we introduce a preorder $\preceq_Q$ on local sketches of a pocket Q, defined as $\Gamma \preceq_Q \Gamma'$ iff $\Gamma(v) \cap T_Q^+ \subseteq \Gamma'(v)$ for all vertices v. Intuitively, maximal elements with respect to this preorder maximize for each vertex, the intersection of its assigned simplex with T_Q^+. We call a local sketch Γ of Q *well-behaved* if it is maximal with respect to $\preceq_Q$, and *interior well-behaved* if it is maximal among all interior local sketches of Q. A similar preorder and notion of well-behaved can be defined for sketches of the entire triangulation, by replacing T_Q^+ by T_{root} in the definition.

The construction. We show in Lemma 5 that for any sketchable pocket Q, we can construct a specific interior well-behaved sketch Λ_Q, and a specific well-behaved sketch Λ_Q^+. Before we can present the proof, we first need to define Λ_Q and Λ_Q^+.

If Q is a trivial pocket, that is, it consists of a single edge e_Q of P, we define

$$\Lambda_Q(v) = \begin{cases} p_i & \text{if } v = c_i, \\ p_j & \text{if } v = c_j, \\ e_Q & \text{otherwise.} \end{cases}$$

For non-trivial pockets Q (that do not consist of a single edge), we will define Λ_Q differently. This definition will rely on the definitions of Λ_L^+ and Λ_R^+ for the child pockets L and R of Q (whose outer triangles T_L^+ and T_R^+ equal the inner triangle T_Q of Q). Therefore, we postpone the definition of Λ_Q for non-trivial pockets until after the definition of Λ_Q^+. Intuitively, Λ_Q^+ pushes those vertices which can be placed anywhere on e_Q out to T_Q^+ if their neighbors allow this and it pushes all remaining vertices as "far out as possible". Formally, Λ_Q^+ is defined in terms of Λ_Q and will hence be defined if and only if Λ_Q is defined:

$$\Lambda_Q^+(v) = \begin{cases} T_Q^+ & \text{if } e_Q \subseteq \Lambda_Q(v) \text{ and } \forall_{(u,v)\in E} \Lambda_Q(u) \cap e_Q \neq \emptyset, \\ \Lambda_Q(v) \cap e_Q & \text{otherwise, if } \Lambda_Q(v) \cap e_Q \neq \emptyset, \\ \Lambda_Q(v) & \text{otherwise.} \end{cases}$$

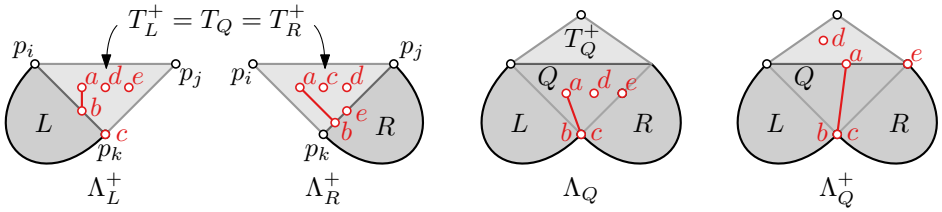


Figure 5 Λ_L^+ and Λ_R^+ are merged into Λ_Q which is then transformed into Λ_Q^+ for a subgraph of G. Vertex c is constrained to p_k in Λ_L^+ and can lie anywhere in T_R^+ in Λ_R^+, hence c is constrained to p_k in both Λ_Q and Λ_Q^+. Vertex a can lie anywhere in $T_L^+ = T_R^+ = T_Q$ in Λ_Q; since a is connected to b it is pushed to the edge (p_i, p_j) in Λ_Q^+ (and not further). Vertex d can also lie anywhere in $T_L^+ = T_R^+ = T_Q$ in Λ_Q; since it has no further restrictions it is pushed all the way to T_Q^+ in Λ_Q^+.

It remains to define Λ_Q for non-trivial pockets. We will define Λ_Q only if Λ_L^+ and Λ_R^+ are defined for both of its child pockets L and R. We attempt to combine Λ_L^+ and Λ_R^+ into a sketch Λ_Q by taking the more restrictive placement for each vertex; here an assignment to T_L^+ or T_R^+ is interpreted as "no placement restriction" (see Figure 5). Potentially, Λ_L^+ and Λ_R^+ restrict the location of a vertex v in such a way that there is no valid placement for v. In such cases, the following definition assigns that vertex to an "undefined" location.

$$\Lambda_Q(v) = \begin{cases} \Lambda_L^+(v) \cap \Lambda_R^+(v) & \text{if } \Lambda_L^+(v) \cap \Lambda_R^+(v) \neq \emptyset, \\ \Lambda_L^+(v) & \text{otherwise, if } T_Q = \Lambda_R^+(v), \\ \Lambda_R^+(v) & \text{otherwise, if } T_Q = \Lambda_L^+(v), \\ \text{undefined} & \text{otherwise.} \end{cases}$$

If the above equation assigns any vertex to an "undefined" location, we say that Λ_Q is *undefined*. In summary, Λ_Q is defined if and only if Λ_L^+ is defined, Λ_R^+ is defined, and the above equation does not assign any vertex to an undefined location. We inductively show that Λ_Q and Λ_Q^+ are defined if and only if the pocket Q is sketchable. Moreover, if they are defined, then Λ_Q and Λ_Q^+ are interior well-behaved and well-behaved sketches, respectively. Lemma 4 shows that if both Λ_Q and Λ_Q^+ are defined, then they are sketches. Lemma 5 shows that if pocket Q is sketchable, then Λ_Q and Λ_Q^+ are both defined and (interior) well-behaved.

We try to construct a sketch Δ for the root triangle T_{root}, which (similar to Λ_Q for a non-trivial pocket Q) combines well-behaved sketches for its child pockets. Where a non-trivial pocket Q has two child pockets, T_{root} has three child pockets A, B, and C (with $T_A^+ = T_B^+ = T_C^+ = T_{\text{root}}$). The equation for Δ is analogous to that of Λ_Q; we say that Δ is defined if and only if all of Λ_A^+, Λ_B^+, and Λ_C^+ are defined, and the following equation does not assign any vertex to an "undefined" location.

$$\Delta(v) = \begin{cases} \Lambda_A^+(v) \cap \Lambda_B^+(v) \cap \Lambda_C^+(v) & \text{if } \Lambda_A^+(v) \cap \Lambda_B^+(v) \cap \Lambda_C^+(v) \neq \emptyset, \\ \Lambda_A^+(v) & \text{otherwise, if } \Lambda_B^+(v) = \Lambda_C^+(v) = T_{\text{root}}, \\ \Lambda_B^+(v) & \text{otherwise, if } \Lambda_A^+(v) = \Lambda_C^+(v) = T_{\text{root}}, \\ \Lambda_C^+(v) & \text{otherwise, if } \Lambda_A^+(v) = \Lambda_B^+(v) = T_{\text{root}}, \\ \text{undefined} & \text{otherwise.} \end{cases}$$

Lemma 6 shows that the triangulation T does not admit a sketch if Δ is undefined. Otherwise, Lemma 4 shows that Δ is a sketch for the triangulation.

▶ **Lemma 4.** *The functions Λ_Q, Λ_Q^+ and Δ are sketches whenever they are defined.*

Proof. Λ_Q, Λ_Q^+ and Δ are sketches if neighboring vertices are assigned to simplices of a common triangle and c_k is assigned to p_k for all k; with $i \leq k \leq j$ in the case of Λ_Q and Λ_Q^+ with $Q = Q_{(p_i,p_j)}$. We prove that Λ_Q and Λ_Q^+ are sketches by structural induction.

First consider Λ_Q for a trivial pocket Q. Λ_Q assigns all vertices to simplices of the triangle containing the edge (p_i, p_j), and assigns $\Lambda_Q(c_i) = p_i$ and $\Lambda_Q(c_j) = p_j$, so Λ_Q is a sketch.

Next, assume that Λ_Q^+ is defined for a pocket Q. By induction, Λ_Q is defined and a sketch. If Λ_Q assigns c_k to p_k, so does Λ_Q^+. If $\Lambda_Q^+(v) = T_Q^+$, all neighbors of v are by definition assigned to simplices of T_Q^+. If $\Lambda_Q^+(v) \neq T_Q^+$, then $\Lambda_Q(v) \supseteq \Lambda_Q^+(v) \neq \emptyset$, so if Λ_Q assigns neighboring vertices to simplices of a common triangle, so does Λ_Q^+. So Λ_Q^+ is a sketch.

Next, assume that Λ_Q is defined for a non-trivial pocket Q. Then Λ_L^+ and Λ_R^+ are sketches for the subpockets L and R of Q with $T_L^+ = T_R^+ = T_Q$. Λ_Q assigns c_k to p_k for all $i \leq k \leq j$. Suppose for a contradiction that Λ_Q assigns two neighboring vertices to simplices that do not share a triangle. Since both Λ_L^+ and Λ_R^+ assign neighboring vertices to simplices of common triangles, there are neighboring vertices u and v such that $\Lambda_Q(u) \not\subseteq \Lambda_L^+(u)$ and $\Lambda_Q(v) \not\subseteq \Lambda_R^+(v)$. So by definition of Λ_Q, we have $\Lambda_L^+(v) \cap \Lambda_R^+(v) = \emptyset$ and $\Lambda_L^+(u) = \Lambda_R^+(v) = T_Q$. Because Λ_L^+ assigns u and v to a common triangle, we have $\Lambda_L^+(u) \cap \Lambda_L^+(v) = T_Q \cap \Lambda_L^+(v) \neq \emptyset$, contradicting that $\Lambda_L^+(v) \cap T_Q = \Lambda_L^+(v) \cap \Lambda_R^+(v) = \emptyset$. Hence Λ_Q is a sketch.

An analogous argument shows that Δ is a sketch if for pockets A, B and C with $T_A^+ = T_B^+ = T_C^+ = T_{\text{root}}$, each of Λ_A^+, Λ_B^+ and Λ_C^+ are sketches, and for all vertices v, we have $T_{\text{root}} \subseteq (\Lambda_B^+(v) \cap \Lambda_C^+(v)) \cup (\Lambda_A^+(v) \cap \Lambda_C^+(v)) \cup (\Lambda_A^+(v) \cap \Lambda_B^+(v))$ or $\Lambda_A^+(v) \cap \Lambda_B^+(v) \cap \Lambda_C^+(v) \neq \emptyset$. ◀

By the following lemma, Λ_Q and Λ_Q^+ are defined if and only if Q has a sketch.

▶ **Lemma 5.** *If a pocket Q is sketchable, then Λ_Q is defined and interior well-behaved, and Λ_Q^+ is defined and well-behaved.*

Proof. We prove this by structural induction along the definitions of Λ_Q and Λ_Q^+.

For the base case, consider Λ_Q for a trivial pocket Q with lid $e_Q = (p_i, p_{i+1})$. As Λ_Q is defined unconditionally, it is a sketch by Lemma 4. Observe that for any interior local sketch Γ of Q, we have $\Gamma(c_i) = p_i$ and $\Gamma(c_{i+1}) = p_{i+1}$. For all vertices $v \notin \{c_i, c_j\}$, we have $\Gamma(v) \cap e_Q \subseteq \Lambda_Q(v)$, so $\Gamma \preceq_Q \Lambda_Q$. That is, Λ_Q is interior well-behaved.

For the inductive step of Λ_Q^+, consider a (not necessarily trivial) sketchable pocket Q with lid $e_Q = (p_i, p_j)$. By induction we may assume that Λ_Q is defined and interior well-behaved. Therefore Λ_Q^+ is defined and a sketch (by Lemma 4). It remains to show that Λ_Q^+ is well-behaved, so for a contradiction suppose that it is not. Then there exists a local sketch Γ of Q and a vertex v for which $\Gamma(v) \cap T_Q^+ \not\subseteq \Lambda_Q^+(v)$. Because Λ_Q does not assign any vertex to simplices outside Q, we have $\Lambda_Q(v) \subseteq Q$, and by definition of Λ_Q^+, we have $\Lambda_Q(v) \cap e_Q \subseteq \Lambda_Q^+(v) \cap T_Q^+$. By interior well-behavedness of Λ_Q and assumption that $\Gamma(v) \cap T_Q^+ \not\subseteq \Lambda_Q^+(v)$, we have $\Gamma(v) \not\subseteq Q$. Therefore, $\Gamma(v) = T_Q^+$ and for all $(u, v) \in E$ we have $\Gamma(u) \cap e_Q \neq \emptyset$. By interior well-behavedness of Λ_Q, we have $\Gamma(v) \cap e_Q = e_Q \subseteq \Lambda_Q(v)$, and for all $(u, v) \in E$, that $\emptyset \neq \Gamma(u) \cap e_Q \subseteq \Lambda_Q(u)$ and hence $\Lambda_Q(u) \cap e_Q \neq \emptyset$. So by definition we have $\Lambda_Q^+(v) = T_Q^+$, contradicting that $\Gamma(v) \cap T_Q^+ \not\subseteq \Lambda_Q^+(v)$, so Λ_Q^+ is well-behaved.

For the inductive step of Λ_Q, suppose that Q is a non-trivial sketchable pocket with lid $e_Q = (p_i, p_j)$. Since Q is sketchable, so are the child pockets L and R of Q (with $T_L^+ = T_R^+ = T_Q$). Inductively, we may assume that Λ_L^+ and Λ_R^+ are well-behaved sketches for L and R. Let Γ_L be the local sketch for L obtained from Γ by replacing $\Gamma(v)$ by $T_L^+ = T_Q$ whenever $\Gamma(v) \not\subseteq L$. Define Γ_R to be the analogous local sketch for R. For the sake of contradiction, assume that Λ_Q is not defined. Then there is some vertex v for which $\Lambda_L^+(v) \cap \Lambda_R^+(v) = \emptyset$ and $T_Q \not\subseteq \Lambda_L^+(v) \cup \Lambda_R^+(v)$. Because $T_Q \not\subseteq \Lambda_L^+(v) \cup \Lambda_R^+(v)$, we have $T_Q \not\subseteq \Gamma_L(v) \cup \Gamma_R(v)$, and therefore $\Gamma(v) \subseteq L$ and $\Gamma(v) \subseteq R$. So $\Gamma(v) \subseteq L \cap R = p_m$, where p_m is the vertex of T_Q not

on the lid of Q. Since $\Gamma(v) \neq \emptyset$, well-behavedness of Λ_L^+ and Λ_R^+ tells us that $p_m \in \Lambda_L^+(v)$ and $p_m \in \Lambda_R^+(v)$. But then $\Lambda_L^+(v) \cap \Lambda_R^+(v) \neq \emptyset$, which is a contradiction, so Λ_Q is defined. To show that Λ_Q is also interior well-behaved, we show that $\Gamma(v) \cap e_Q \subseteq \Lambda_Q(v)$ for all $v \in V$. By well-behavedness of Λ_L^+ and Λ_R^+, we have $\Gamma(v) \cap e_Q \subseteq \Lambda_L^+(v)$ and $\Gamma(v) \cap e_Q \subseteq \Lambda_R^+(v)$. So $\Gamma(v) \cap e_Q \subseteq \Lambda_L^+(v) \cap \Lambda_R^+(v)$, and by definition of Λ_Q we have $\Lambda_L^+(v) \cap \Lambda_R^+(v) \subseteq \Lambda_Q(v)$, and hence $\Gamma(v) \cap e_Q \subseteq \Lambda_Q(v)$. So Λ_Q is interior well-behaved. ◀

Similarly, we can show for a given $\mathcal{T}$ of a polygon P, that if $\mathcal{T}$ has a sketch, then Δ is defined and a well-behaved sketch.

▶ **Lemma 6.** *If there exists a sketch for a given triangulation $\mathcal{T}$ of a simple polygon P, then Δ is defined and well-behaved.*

The following two corollaries summarize that the existence of a sketch is equivalent to the existence of a well-behaved sketch, both for pockets Q and for a complete triangulation $\mathcal{T}$.

▶ **Corollary 7.** *Λ_Q and Λ_Q^+ are defined if and only if Q has a sketch.*

▶ **Corollary 8.** *Sketch Δ is defined if and only if the triangulation $\mathcal{T}$ has a sketch.*

Computing triangulation-respecting drawings. Any sketch implies a drawing that places vertices anywhere in their assigned simplex. Conversely, any triangulation-respecting drawing implies a sketch that assigns vertices to the corresponding simplex. The definition of Δ hence directly results in a $\mathcal{O}(t|V||E|)$-time algorithm both to decide the existence of, and to compute a triangulation-respecting drawing. We can improve this running time to linear.

▶ **Theorem 9.** *There is a linear-time algorithm to decide if (G, C) has a triangulation-respecting drawing for a simple polygon P with fixed triangulation $\mathcal{T}$; the same algorithm also constructs a drawing if one exists.*

4 Planar triangulation-respecting drawings

We are given the same input as in Section 3, namely an instance (G, C) consisting of a graph G with n vertices and a cycle C with t vertices, and a simple polygon P with t vertices together with an arbitrary triangulation $\mathcal{T}$ of P. In addition, we assume that the instance (G, C) is planar, that is, G has a planar drawing $\mathcal{D}$ with C on the outer face. Note that $\mathcal{D}$ does not necessarily map vertices of C to vertices of P.

Analogously to Section 3, we can ask the following question: is there a planar drawing of (G, C) that respects both P and $\mathcal{T}$? The answer to this question is often "no", even when both triangulation-respecting drawings and planar polygon-respecting drawings exist. Consider, for example, Figure 6: a planar triangulation-respecting drawing for this combination

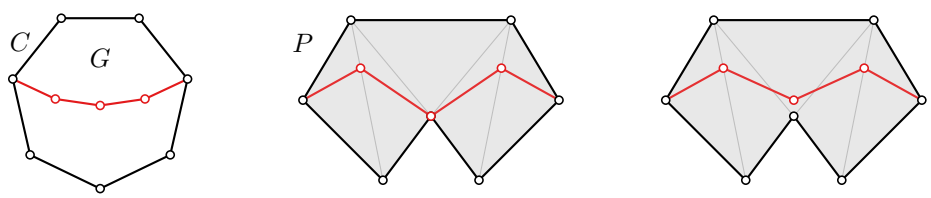


Figure 6 Left: A planar instance. Center: a triangulation-respecting drawing in which two vertices coincide. Right: a perturbed drawing that is planar but not triangulation-respecting.

of (G, C), P, and $\mathcal{T}$ does not exist; any drawing inside P either places two vertices on top of each other, or edges cross edges of the triangulation. Still, triangulation-respecting drawings are a useful tool for our final goal of constructing planar polygon-respecting drawings. The triangulation-respecting drawing of Figure 6 can be perturbed infinitesimally to obtain a planar polygon-respecting drawing (that is not triangulation-respecting). In this section, we show that if a planar instance (G, C) has a triangulation-respecting drawing, then it also has a *weakly-planar triangulation-respecting* one: a triangulation-respecting drawing that is planar and polygon-respecting after infinitesimal perturbation (moving vertices to simplices of $\mathcal{T}$ that contain their original location.) Hence, the algorithm in Section 3 can decide for a planar instance (G, C) whether there is a weakly-planar triangulation-respecting drawing.

Let $\mathcal{D}$ be a planar drawing of (G, C). We call the triple $(G, C, \mathcal{D})$ a *plane instance*. We say that a weakly-planar triangulation-respecting drawing $\mathcal{W}$ *accommodates* $(\mathcal{D}, \mathcal{T})$ if there exists a planar polygon-respecting infinitesimal perturbation $\widetilde{\mathcal{W}}$ of $\mathcal{W}$ that is isotopic to $\mathcal{D}$ in the plane. That is, one can be continuously deformed into the other without introducing self-intersections. We now construct a weakly-planar triangulation-respecting drawing that accommodates $(\mathcal{D}, \mathcal{T})$. A plane instance $(G, C, \mathcal{D})$ is *sketchable* if (G, C) has a sketch (for $\mathcal{T}$). Recall that a sketch does not have a notion of planarity. However, we show in Theorem 11 that any sketchable plane instance $(G, C, \mathcal{D})$ has a drawing $\mathcal{W}$ which accommodates $(\mathcal{D}, \mathcal{T})$.

Minimal plane instances. We show how to transform any plane instance $(G, C, \mathcal{D})$ into a *minimal* plane instance, while preserving its sketchability; details can be found in the full version. First of all, we carefully triangulate $(G, C, \mathcal{D})$, so as not to influence sketchability. If all faces of $\mathcal{D}$ interior to C are triangles, we call $(G, C, \mathcal{D})$ a *triangulated instance*. A triangulation of a plane instance $(G, C, \mathcal{D})$ is a triangulated instance $(G', C, \mathcal{D}')$ such that G' is a supergraph of G (with potentially additional vertices) and $\mathcal{D}$ is the restriction of $\mathcal{D}'$ to G. Second, we remove the interior of all separating triangles. If G does not have any separating triangles, then we contract any edge not on C that preserves sketchability and remove the interior of any separating triangles this edge contraction might create. If no further simplifications are possible, we call $(G, C, \mathcal{D})$ minimal.

▶ **Lemma 10.** *Every sketchable minimal plane instance $(G, C, \mathcal{D})$ has a drawing that accommodates $(\mathcal{D}, \mathcal{T})$.*

Proof. Consider a pocket $Q = Q_{(p_i, p_j)}$. We claim that if $\Lambda^+_Q(v) = p_k$ for some $k \in [i, j]$, then $v = c_k$, and moreover that if $\Lambda^+_Q(v) = e_Q$, then Q does not consist of a single edge and v is a neighbor of c_m, where p_m is the third vertex of T_Q. If Q consists of a single edge, then the claim clearly holds. If Q does not consist of a single edge, we have by induction that the claim holds for the subpockets $L = Q_{(p_i, p_m)}$ and $R = Q_{(p_m, p_j)}$. If $\Lambda^+_L(v) = e_L$ for some v, then we claim that the instance is not minimal. Let p_l be the third vertex of T_L, and without loss of generality assume that v is the most counter-clockwise neighbor (according to $\mathcal{D}$) of c_l for which $\Lambda^+_L(v) = e_L$ (see Figure 7). Then the (triangular) face counter-clockwise of

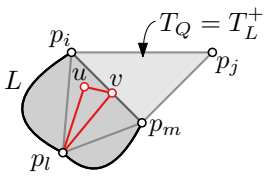


Figure 7 The vertex u must be assigned to p_i or p_m. In both cases an edge can be contracted while preserving the existence of a sketch.

edge (c_l, v) is a triangle whose third vertex u does not have $\Lambda_L^+(u) = e_L$. Since u has c_l and v as neighbors, $\Lambda_L^+(u)$ is a simplex of T_L, but not e_L, so by definition of Λ_L^+ it is a vertex of T_L. By induction, u is therefore c_m, c_l, or c_i. Because u is a neighbor of c_l, u itself is not c_l. We argue that we can contract an edge while preserving sketchability, contradicting minimality.

First consider the case where $u = c_m$. Then (c_l, c_m) divides G into two subgraphs, to the left and to the right of (c_l, c_m). We obtain a new sketch by reassigning all vertices of the right subgraph that are placed outside the pocket with lid (p_l, p_m) to that lid. By construction, the triangle c_l, c_m, v lies right of (c_l, c_m). Then v and its neighbors are reassigned to p_l, p_m, or (p_l, p_m), so contracting the edge (u, v) maintains sketchability, contradicting minimality.

Now consider the case where $u = c_i$. If $\Lambda_Q^+(u) = \Lambda_Q^+(v)$, it is clear that we can contract the edge and preserve a sketch. So since $\Lambda_Q^+(v)$ is either p_i or p_m, we have $\Lambda_Q^+(v) = p_m$, but then $\Lambda_R^+(v)$ is not T_Q, so either $\Lambda_R^+(v) = p_m$ or $\Lambda_R^+(v) = e_R$. The first case implies $v = c_m$ and hence contradicts $\Lambda_L^+(v) = e_L$. In the second case, the inductive hypothesis implies that v is a neighbor of the third vertex p_r of the triangle T_R. Now consider the subgraph of G that is right of the path consisting of the edges (p_l, v) and (v, p_r). Any of its vertices that is assigned outside L and R can be assigned to p_m, maintaining a sketch. There exists a path from c_m to v that avoids c_l and c_r, and the last edge of this path can be contracted, contradicting minimality. So $\Lambda_L^+(v) \neq e_L$ and symmetrically $\Lambda_R^+(v) \neq e_R$. By definition, Λ_Q^+ assigns (for $k \in [i, j]$) only c_k to p_k, and only neighbors of c_m to e_Q, so the claim holds.

Therefore, in the sketch Δ, all vertices other than those of C are assigned to edges of T_{root}, or T_{root} itself. If there is such a vertex not on C, then contracting an edge between C and $G \setminus C$ yields an instance with a sketch, contradicting minimality. So all vertices in a minimal instance lie on C. Because G is triangulated, this means that its edges coincide with those of the triangulation of P. So the instance clearly has an accommodating drawing. ◀

The proof of Theorem 11 shows that the accommodating drawing of the minimal instance obtained from the simplification procedure (if that instance is sketchable) can be extended to be an accommodating drawing for the original instance, by undoing the simplification steps.

▶ **Theorem 11.** *A plane instance $(G, C, \mathcal{D})$ has a drawing that accommodates $(\mathcal{D}, \mathcal{T})$ if and only if $(G, C, \mathcal{D})$ is sketchable.*

The algorithm implied by Theorem 9 can check in linear time if a plane instance $(G, C, \mathcal{D})$ has a sketch and via Theorem 11 the same algorithm can decide in linear time if $(G, C, \mathcal{D})$ has an accommodating drawing. This drawing can be constructed in polynomial time, following the (polynomial number of) steps in the minimization procedure.

5 Sufficient conditions for polygon-universality

In Section 2 we proved that the Pair and Triple Conditions are necessary for an instance (G, C) to be polygon-universal. Here we show, using triangulation-respecting drawings, that these two conditions are sufficient as well. In Sections 3 and 4 we argued that an instance (G, C) has a triangulation-respecting drawing for a triangulation $\mathcal{T}$ of P if and only if it has a sketch for $\mathcal{T}$; we also gave an algorithm that tests whether such a sketch exists. Below we show that if the Pair and Triple Conditions are satisfied for an instance (G, C) then it has a sketch for any triangulation $\mathcal{T}$. We do so by examining the testing algorithm more closely.

We first show that the Pair Condition alone already implies that each pocket has a sketch. The Triple Condition then allows us to combine sketches at the root T_{root} of $\mathcal{T}$. Denote by $Q_e = Q_{(p_i, p_j)}$ the pocket with lid $e = (p_i, p_j)$. We want to determine whether an individual vertex can be drawn outside a pocket. Definition 12 relates the position of a vertex in a sketch of a pocket to its distance to points on the cycle, see Figure 8.

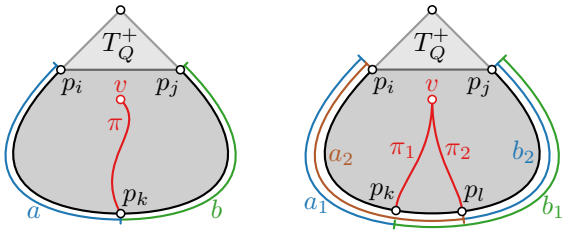


Figure 8 Cases for Definition 12. Left: we have $\pi \leq a$ and $\pi \leq b$. Right: we have $\pi_1 \leq a_1 + 1$, $\pi_1 \leq b_1 - 1$, $\pi_2 \leq a_2 - 1$, and $\pi_2 \leq a_2 + 1$.

▶ **Definition 12.** *Vertex v is* pulled *by pocket $Q_{(p_i,p_j)}$ if and only if either of the following two conditions hold:*

1. *for some $k \in \{i, \ldots, j\}$, we have $d_G(v, c_k) \leq \min(k - i, j - k)$;*
2. *for some $k, l \in \{i, \ldots, j\}$, we have $d_G(v, c_k) \leq \min(k - i + 1, j - k - 1)$ and $d_G(v, c_l) \leq \min(l - i - 1, j - l + 1)$.*

Let Q be a pocket and let v be an arbitrary vertex. Either the well-behaved sketch Λ_Q^+ of Q places v outside of Q or Lemma 13 characterizes where in Q vertex v "is stuck".

▶ **Lemma 13.** *Let v be a vertex and $Q = Q_{(p_i,p_j)}$ with $i < j < i + t$ be a sketchable pocket.*

1. *If $p_j \notin \Lambda_Q^+(v)$, then $v = c_{j-1}$ or there exists a triangle in Q with vertices p_a, p_b, p_j and $i \leq a < b < j$ such that for pocket $Q' = Q_{(p_a,p_b)}$, $\Lambda_{Q'}^+(v) \neq T_{Q'}^+$.*
2. *If $p_i \notin \Lambda_Q^+(v)$, then $v = c_{i+1}$ or there exists a triangle in Q with vertices p_i, p_a, p_b and $i < a < b \leq j$ such that for pocket $Q' = Q_{(p_a,p_b)}$, $\Lambda_{Q'}^+(v) \neq T_{Q'}^+$.*

Proof. Statements (1) and (2) can be proved using symmetric arguments, so we prove only statement (1). We proceed by induction on the size of Q. If Q consists of the single edge (p_i, p_j), then $i = j - 1$, and if $p_j \notin \Lambda_Q^+(v)$, then $v = c_i = c_{j-1}$. So assume that Q does not consist of a single edge. Let p_m with $i < m < j$ be the third vertex of T_Q and let $L = Q_{(p_i,p_m)}$ and $R = Q_{(p_m,p_j)}$. If $p_j \notin \Lambda_Q^+(v)$, then $\Lambda_L^+(v)$ or $\Lambda_R^+(v)$ does not contain p_j. If $p_j \notin \Lambda_R^+(v)$, then by induction we are done. So assume that $p_j \in \Lambda_R^+(v)$ and hence that $p_j \notin \Lambda_L^+(v)$. This means that $\Lambda_L^+(v) \neq T_L^+$, completing the proof. ◀

Lemma 14 and 15 relate the characterization in Lemma 13, which uses the well-behaved sketch Λ_Q^+, to the requirements on graph and cycle distances expressed in Definition 12. Lemma 14 covers the first condition of Definition 12, while Lemma 15 covers the remainder.

▶ **Lemma 14.** *Assume that pocket $Q = Q_{(p_i,p_j)}$ with $i < j < i + t$ admits a sketch. If $\Lambda_Q^+(v) \neq T_Q^+$, then v is pulled by Q.*

Proof. If $\Lambda_Q^+(v) \neq T_Q^+$, then either $e_Q \not\subseteq \Lambda_Q(v)$ or for some neighbor u of v, $\Lambda_Q(u)$ does not intersect e_Q. We proceed by induction on the size of Q.

***Q* is trivial.** If Q is trivial, it consists of one edge e_Q, and $\Lambda_Q(u)$ intersects e_Q for all u. By assumption that $\Lambda_Q^+(v) \neq T_Q^+$, we have $e_Q \not\subseteq \Lambda_Q(v)$ by definition of Λ_Q^+, so $v = c_i$ or $v = c_j$. Hence, $d_G(v, c_k) = 0 \leq \min\{k - i, j - k\} = 0$ for some $k \in \{i, j\}$, so v is pulled by Q.

***Q* is non-trivial.** Next, suppose that Q is non-trivial, and thus contains the triangle T_Q. Let p_m (with $i < m < j$) be the third vertex of T_Q and let $L = Q_{(p_i,p_m)}$ and $R = Q_{(p_m,p_j)}$ be the two subpockets of Q with $T_L^+ = T_R^+ = T_Q$. If $e_Q \not\subseteq \Lambda_Q(v)$, then $T_Q \neq \Lambda_L^+(v)$

or $T_Q \neq \Lambda_R^+(v)$, so by induction v is pulled by L or R, and hence also by Q. So assume that $e_Q \subseteq \Lambda_Q(v)$ and there exists some neighbor u of v for which $\Lambda_Q(u)$ does not intersect e_Q. It follows by construction that $\Lambda_L^+(v) = \Lambda_R^+(v) = T_Q$. Because u is assigned to a simplex of the same triangle as v, we have $\Lambda_L^+(u) \subseteq T_Q$ and $\Lambda_R^+(u) \subseteq T_Q$, so $\Lambda_Q(u) \subseteq T_Q$. Since $\Lambda_Q(u)$ does not intersect e_Q, we have $\Lambda_Q(u) = p_m$ and hence $\Lambda_L^+(u) \cap \Lambda_R^+(u) = p_m$. So either $\Lambda_L^+(u) = e_L$ and $\Lambda_R^+(u) = e_R$, or $\Lambda_L^+(u)$ or $\Lambda_R^+(u)$ is p_m. We consider these cases separately.

Edge case. Suppose that $\Lambda_L^+(u) = e_L$ and $\Lambda_R^+(u) = e_R$. Then $m \in [i+2, j-2]$ and by induction u is pulled by both L and R. We distinguish three cases depending on what causes u to be pulled by L and R, and show in each case that v is pulled by Q.

1. If there exists some $l \in [i, m]$ with $d_G(u, c_l) \leq \min(l - i - 1, m - l + 1)$, then

$$\begin{aligned} d_G(v, c_l) &\leq d_G(u, c_l) + 1 \\ &\leq \min(l - i - 1, m - l + 1) + 1 \\ &\leq \min(l - i, j - l), \end{aligned}$$

   so v is pulled by Q.
2. Symmetrically, v is pulled by Q if $d_G(u, c_k) \leq \min(k - m + 1, j - k - 1)$ for some $k \in [m, j]$.
3. In the remaining case, there exist $k \in [i, m]$ and $l \in [m, j]$ with $d_G(u, c_k) \leq \min(k - i, m - k)$ and $d_G(u, c_l) \leq \min(l - m, j - l)$. Therefore

$$\begin{aligned} d_G(v, c_k) &\leq d_G(u, c_k) + 1 \\ &\leq \min(k - i, m - k) + 1 \\ &\leq \min(k - i + 1, m - k + 1) \\ &\leq \min(k - i + 1, j - k - 1) \end{aligned}$$

   and symmetrically $d_G(v, c_l) \leq \min(l - i - 1, j - l + 1)$, so v is pulled by Q.

Corner case. Assume that $\Lambda_L^+(u) = p_m$ (the case $\Lambda_R^+(u) = p_m$ is symmetric). Then $p_i \notin \Lambda_L^+(u)$, so by Lemma 13, we have either $u = c_{i+1}$ or there exists some pocket $Q' = Q_{(p_a, p_b)}$ with $i < a < b \leq m$ such that $\Lambda_{Q'}^+(u) \neq T_{Q'}^+$. If $u = c_{i+1}$, then for $k = i + 1$ we have $d_G(v, c_k) \leq d_G(u, c_k) + 1 \leq 1 \leq \min(k - i, j - k)$ in which case v is pulled by Q. Otherwise, u is by induction pulled by some pocket $Q_{(p_a, p_b)}$ with $i < a < b \leq m < j$, and since $d_G(v, u) \leq 1$, the triangle inequality shows that v is pulled by $Q_{(p_i, p_j)}$.

By induction, v is pulled by Q whenever $\Lambda_Q^+(v) \neq T_Q^+$. ◀

▶ **Lemma 15.** *Assume that pocket $Q = Q_{(p_i, p_j)}$ with $i < j < i + t$ admits a sketch. If v is pulled by Q but there exists no $k \in [i, j]$ such that $d_G(v, c_k) \leq \min(k - i, j - k)$, then there exist $k, o, l \in [i, j]$ such that $k < o < l$ and a vertex $x \neq v$ with $d_G(x, c_k) \leq \min(k - i, o - k) \leq \min(k - i + 1, j - k - 1) - d_G(x, v)$ and $d_G(x, c_l) \leq \min(l - o, j - l) \leq \min(l - i - 1, j - l + 1) - d_G(x, v)$.*

Lemma 16 ties together our preceding arguments to show that if a pocket has no sketch, then the Pair Condition is violated. This directly implies Corollary 17.

▶ **Lemma 16.** *If pocket $Q = Q_{(p_i, p_j)}$ with $i < j < i + t$ has no sketch, then there exist $i \leq k \leq l \leq j$ such that $d_G(c_k, c_l) < d_C(c_k, c_l)$.*

▶ **Corollary 17.** *If the Pair Condition is satisfied, then all pockets have a sketch.*

We now established that the Pair Condition implies that each pocket has a sketch. If additionally the Triple Condition is satisfied, then Theorem 18 shows that we can combine the sketches for the three pockets, whose lids are the edges of the root triangle T_{root}, to obtain a sketch, and hence a triangulation-respecting drawing, for (G, C).

▶ **Theorem 18.** *If an instance (G, C) satisfies the Pair and Triple Conditions, then (G, C) has a triangulation-respecting drawing for any triangulation $\mathcal{T}$ of any simple polygon P.*

▶ **Corollary 19.** *If a plane instance $(G, C, \mathcal{D})$ satisfies the Pair and Triple Conditions, then (G, C) has a drawing that accommodates $(\mathcal{D}, \mathcal{T})$ for any triangulation $\mathcal{T}$ of any simple polygon P.*

6 Discussion and conclusion

We have characterized the (planar) polygon-universal graphs (G, C) by means of simple combinatorial conditions involving (graph-theoretic) distances along the cycle C and in the graph G. In particular, this shows that, even though the recognition of polygon-universal graphs most naturally lies in $\forall\exists\mathbb{R}$, it can in fact be tested in polynomial time, by explicitly checking the Pair and the Triple Conditions. Our main open question concerns the restriction to simple polygons without holes. Can a similar characterization be achieved in the presence of holes? Or is the polygon-universality problem for simple polygons with holes $\forall\exists\mathbb{R}$-complete?

Another interesting question concerns the running time for recognizing polygon-universal graphs. Testing the Pair and Triple Conditions naively requires at least $\Omega(n^3)$ time. On the other hand, at least in the non-planar case, given (G, C) and a polygon P with arbitrary triangulation T, we can in linear time either find an extension or a violation of the Pair/Triple Condition, which shows that the instance is not polygon-universal. (Recall that a polygon-extension for P might exist, though not one that respects T, see Figure 2). For planar instances, the contraction to minimal instances causes an additional linear factor in the running time. Can (planar) polygon-universality be tested in $o(n^2)$ time?

References

1. Patrizio Angelini, Giuseppe Di Battista, Fabrizio Frati, Vít Jelínek, Jan Kratochvíl, Maurizio Patrignani, and Ignaz Rutter. Testing planarity of partially embedded graphs. *ACM Transactions on Algorithms*, 11(4):32:1–32:42, 2015. doi:10.1145/2629341.
2. Patrizio Angelini, Philipp Kindermann, Andre Löffler, Lena Schlipf, and Antonios Symvonis. One-bend drawings of outerplanar graphs inside simple polygons. In *Abstr. 36th European Workshop on Computational Geometry*, pages 70:1–70:6, 2020.
3. John Canny. Some algebraic and geometric computations in PSPACE. In *Proc. 20th Symposium on Theory of Computing*, page 460–467, 1988. doi:10.1145/62212.62257.
4. Erin W. Chambers, David Eppstein, Michael T. Goodrich, and Maarten Löffler. Drawing graphs in the plane with a prescribed outer face and polynomial area. *Journal of Graph Algorithms and Applications*, 16(2):243–259, 2012. doi:10.7155/jgaa.00257.
5. Michael G. Dobbins, Linda Kleist, Tillmann Miltzow, and Paweł Rzążewski. $\forall\exists\mathbb{R}$-completeness and area-universality. In *Proc. 44th International Workshop on Graph-Theoretic Concepts in Computer Science*, LNCS 11159, pages 164–175, 2018. doi:10.1007/978-3-030-00256-5_14.
6. Christian A. Duncan, Michael T. Goodrich, and Stephen G. Kobourov. Planar drawings of higher-genus graphs. *Journal of Graph Algorithms and Applications*, 15(1):7–32, 2011. doi:10.1007/978-3-642-11805-0_7.
7. Seok-Hee Hong and Hiroshi Nagamochi. Convex drawings of graphs with non-convex boundary constraints. *Discrete Applied Mathematics*, 156(12):2368–2380, 2008. doi:10.1016/j.dam.2007.10.012.

8 Vít Jelínek, Jan Kratochvíl, and Ignaz Rutter. A Kuratowski-type theorem for planarity of partially embedded graphs. *Computational Geometry*, 46(4):466–492, 2013. doi:10.1016/j.comgeo.2012.07.005.

9 Anna Lubiw, Tillmann Miltzow, and Debajyoti Mondal. The complexity of drawing a graph in a polygonal region. In *Proc. 26th International Symposium on Graph Drawing and Network Visualization*, LNCS 11282, pages 387–401, 2018. doi:10.1007/978-3-030-04414-5_28.

10 Tamara Mchedlidze, Martin Nöllenburg, and Ignaz Rutter. Extending convex partial drawings of graphs. *Algorithmica*, 76(1):47–67, 2016. doi:10.1007/s00453-015-0018-6.

11 Tamara Mchedlidze and Jérôme Urhausen. β-stars or on extending a drawing of a connected subgraph. In *Proc. 26th International Symposium on Graph Drawing and Network Visualization*, LNCS 11282, pages 416–429, 2018. doi:10.1007/978-3-030-04414-5_30.

12 Marcus Schaefer and Daniel Štefankovič. Fixed points, nash equilibria, and the existential theory of the reals. *Theory of Computing Systems*, 60:172–193, 2017. doi:10.1007/s00224-015-9662-0.

13 William T. Tutte. How to draw a graph. *Proceedings of the London Mathematical Society*, 13(3):743–768, 1963. doi:10.1112/plms/s3-13.1.743.

On Rich Points and Incidences with Restricted Sets of Lines in 3-Space

Micha Sharir
School of Computer Science, Tel Aviv University, Israel

Noam Solomon
School of Computer Science, Tel Aviv University, Israel

Abstract

Let L be a set of n lines in $\mathbb{R}^3$ that is contained, when represented as points in the four-dimensional Plücker space of lines in $\mathbb{R}^3$, in an irreducible variety T of constant degree which is *non-degenerate* with respect to L (see below). We show:

(1) If T is two-dimensional, the number of r-rich points (points incident to at least r lines of L) is $O(n^{4/3+\varepsilon}/r^2)$, for $r \geqslant 3$ and for any $\varepsilon > 0$, and, if at most $n^{1/3}$ lines of L lie on any common regulus, there are at most $O(n^{4/3+\varepsilon})$ 2-rich points. For r larger than some sufficiently large constant, the number of r-rich points is also $O(n/r)$.
As an application, we deduce (with an ε-loss in the exponent) the bound obtained by Pach and de Zeeuw [16] on the number of distinct distances determined by n points on an irreducible algebraic curve of constant degree in the plane that is not a line nor a circle.

(2) If T is two-dimensional, the number of incidences between L and a set of m points in $\mathbb{R}^3$ is $O(m+n)$.

(3) If T is three-dimensional and nonlinear, the number of incidences between L and a set of m points in $\mathbb{R}^3$ is $O\left(m^{3/5}n^{3/5} + (m^{11/15}n^{2/5} + m^{1/3}n^{2/3})s^{1/3} + m + n\right)$, provided that no plane contains more than s of the points. When $s = O(\min\{n^{3/5}/m^{2/5}, m^{1/2}\})$, the bound becomes $O(m^{3/5}n^{3/5} + m + n)$.

As an application, we prove that the number of incidences between m points and n lines in $\mathbb{R}^4$ contained in a quadratic hypersurface (which does not contain a hyperplane) is $O(m^{3/5}n^{3/5} + m + n)$.

The proofs use, in addition to various tools from algebraic geometry, recent bounds on the number of incidences between points and algebraic curves in the plane.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Lines in space, Rich points, Polynomial partitioning, Incidences

Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.56

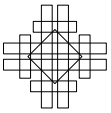
Related Version *Full Version*: https://arXiv.org/abs/2012.11913

Funding *Micha Sharir*: Work partially supported by ISF Grant 260/18, by grant 1367/2016 from the German-Israeli Science Foundation (GIF), and by Blavatnik Research Fund in Computer Science at Tel Aviv University.

Acknowledgements We deeply thank Larry Guth for helpful interaction on this work.

1 Introduction

The setup: Incidences between a set of points and a restricted set of lines in $\mathbb{R}^3$. Let P be a set of m points and L a set of n lines in $\mathbb{R}^3$. We consider the problem of obtaining sharp incidence bounds between the points of P and the lines of L, when the lines of L, considered as points in the four-dimensional Plücker space of lines in $\mathbb{R}^3$, are restricted to lie on a two- or three-dimensional constant-degree algebraic variety T. The topic of incidences between points and lines is a fundamental topic in incidence geometry, significantly boosted



since Guth and Katz's seminal work [11] on point-line incidences in $\mathbb{R}^3$. Instead of asking for a bound on the number of incidences between points and lines, we can ask, for each $r \geqslant 3$, for a bound on the number of *r-rich points* in a set of lines, which are the points that are incident to at least r of the lines. As it turns out, the two questions are equivalent. The related, and finer problem of bounding the number of 2-rich points, determined by a set of n lines in $\mathbb{R}^3$, studied in [11], is also discussed in this paper, under the restricted setup considered here. Building on recent works of Sharir and Zahl [25] and Zahl [29], we are able to improve Guth and Katz's point-line incidence bounds when the lines in L are restricted to lie on a two- or three-dimensional variety T in the Plücker space.

Background: Points and curves, the planar case. The study of incidences between points and curves has a rich history, starting with the simplest instance of points and lines in the plane, where we have (see also [4, 27]):

▶ **Theorem 1** (Szemerédi and Trotter [28]). *The maximum number of incidences between m points and n lines in the plane is $\Theta(m^{2/3}n^{2/3} + m + n)$.*

In fact, an equivalent formulation of Szemerédi-Trotter theorem asserts that, given n lines in the plane, the number of points that are incident to at least r of the lines, for any parameter $2 \leqslant r \leqslant n$, which we call *r-rich points* and denote the set of these points by $P_{\geqslant r}(L)$, satisfies

$$|P_{\geqslant r}(L)| = O\left(\frac{n^2}{r^3} + \frac{n}{r}\right). \tag{1}$$

Still in the plane, Pach and Sharir [17] extended this bound to incidence bounds between points and curves with k *degrees of freedom*, namely, for each set of k distinct points, there are only $\mu = O(1)$ curves that pass through all of them, and each pair of curves intersect in at most μ points; μ is called the *multiplicity* (of the degrees of freedom). Here is their result, tailored to the case of algebraic curves.

▶ **Theorem 2** (Pach and Sharir [17]). *Let P be a set of m points in $\mathbb{R}^2$ and let $\mathcal{C}$ be a set of n bounded-degree algebraic curves in $\mathbb{R}^2$ with k degrees of freedom and with multiplicity μ. Then (where the constant of proportionality depends on k and μ)*

$$I(P,\mathcal{C}) = O\left(m^{\frac{k}{2k-1}}n^{\frac{2k-2}{2k-1}} + m + n\right).$$

Except for the case $k = 2$ (lines have two degrees of freedom), the bound is not known, and is strongly suspected not to be tight in the worst case (see [1, 2, 15] for an improvement for the case of circles and similar curves).

Recently, Sharir and Zahl [25] have considered general families of constant-degree algebraic curves in the plane that belong to an *s-dimensional family of curves*. This means that each curve in such a family can be represented by a constant number of real parameters, so that, in this parametric space, the points representing the curves lie in an s-dimensional algebraic variety $\mathcal{F}$ of some constant degree (the so-called "complexity" of $\mathcal{F}$). See [25] for details.

▶ **Theorem 3** (Sharir and Zahl [25]). *Let $\mathcal{C}$ be a set of n algebraic plane curves that belong to an s-dimensional family $\mathcal{F}$ of curves of maximum constant degree E, no two of which share a common irreducible component, and let P be a set of m points in the plane. Then, for any $\varepsilon > 0$, the number $I(P,\mathcal{C})$ of incidences between the points of P and the curves of $\mathcal{C}$ satisfies*

$$I(P,\mathcal{C}) = O\Big(m^{\frac{2s}{5s-4}}n^{\frac{5s-6}{5s-4}+\varepsilon} + m^{2/3}n^{2/3} + m + n\Big),$$

where the constant of proportionality depends on ε, s, E, and the complexity of the family $\mathcal{F}$.

Except for the factor $O(n^{\varepsilon})$, this is a significant improvement over the bound in Theorem 2 (for $s \geqslant 3$) when $\mathcal{C}$ has $k = s$ degrees of freedom (as it often does).

Incidences with lines in three dimensions. The groundbreaking work of Guth and Katz [11] implies[1] the sharper bound $O(m^{1/2}n^{3/4} + m^{2/3}n^{1/3}q^{1/3} + m + n)$ on the number of incidences between m points and n lines in $\mathbb{R}^3$, provided that no plane contains more than q of the given lines. We use the following variant (see the full version [21] for the proof).

▶ **Theorem 4.** *Let P be a set of m points in $\mathbb{R}^3$, and let L be a set of n lines in $\mathbb{R}^3$, so that no 2-flat contains more than s points of P. Then*

$$I(P, L) = O(m^{1/2}n^{3/4} + m^{1/3}n^{2/3}s^{1/3} + m + n).$$

Plugging the bound of Theorem 4 into the proof of [23, Theorem 1.3(a)], we get

▶ **Theorem 5.** *Let P be a set of m points and L a set of n lines in $\mathbb{R}^d$, for $d \geqslant 3$, so that all the points and lines lie in a two-dimensional algebraic variety V of degree D that does not contain any 2-flat, and so that no 2-flat contains more than s points of P. Then*

$$I(P, L) = O(m^{1/2}n^{1/2}D^{1/2} + m^{1/3}D^{4/3}s^{1/3} + m + n).$$

Guth and Katz's work has lead to many recent works on incidences between points and lines or other curves in three and higher dimensions; see [3, 12, 19, 22, 23, 20] for a sample.

Of particular significance is the recent work of Guth and Zahl [12] on the number of 2-rich points in a collection of algebraic curves of constant degree, namely, points incident to at least two of the given curves, which extends Guth and Katz's bound of $O(n^{3/2})$, obtained for the case of lines, when no plane or regulus contains more than $O(n^{1/2})$ lines [11]. The extension requires analogous (but stricter) restrictive assumptions (concerning surfaces that are doubly ruled by the given family of curves).

Our new bounds require the extension to three dimensions of the notions of having k degrees of freedom and of being an s-dimensional family of curves. The definitions of these concepts, as given above for the planar case, extend, basically verbatim, to three (or higher) dimensions, but, even in typical situations, these two concepts do not coincide anymore.

Our results. We obtain improved incidence bounds when the lines of L, as points in Plücker space, lie on a two- or three-dimensional variety T. When T is two-dimensional and non-planar, the number of r-rich points is $O(n^{4/3+\varepsilon}/r^2)$, for $r \geqslant 3$ and for any $\varepsilon > 0$, and, if at most $n^{1/3}$ lines of L lie on any common regulus, there are at most $O(n^{4/3+\varepsilon})$ 2-rich points. For r larger than some sufficiently large constant, the number of r-rich points is also $O(n/r)$, which is a better bound when $r = O(n^{1/3})$. These bounds improve significantly, for the restricted context at hand, the bound $O(n^{3/2}/r^2)$ due to Guth and Katz [11] (which holds when no plane or regulus contains more than $O(n^{1/2})$ lines). Moreover, the number of incidences between L and a set of m points in $\mathbb{R}^3$ is $O(m+n)$, again a significant improvement, in our context, over the previous bound in [11].

As an application, we show that the number of distinct distances determined by n points on an irreducible algebraic curve of constant degree in the plane that is not a line nor a circle, is $\Omega(n^{4/3-\varepsilon})$, for any $\varepsilon > 0$, which is (with an ε-loss in the exponent) the bound obtained by Pach and de Zeeuw [16].

[1] This bound is not explicitly stated in [11], but it readily follows from the analysis given there, and by now it is generally attributed to that work.

If T is three-dimensional and nonlinear, the number of incidences between L and a set of m points in $\mathbb{R}^3$ is $O\left(m^{3/5}n^{3/5} + (m^{11/15}n^{2/5} + m^{1/3}n^{2/3})s^{1/3} + m + n\right)$, provided that no plane contains more than s of the points. When $s = O(\min\{n^{3/5}/m^{2/5}, m^{1/2}\})$, the bound becomes $O(m^{3/5}n^{3/5} + m + n)$.

An interesting novel feature of our results is that, like Theorem 4, it is obtained under an assumption that restricts the number of *points* that can lie on a common plane (instead of restricting the number of coplanar lines in the previous studies). Very few earlier works have used this kind of restriction; see Elekes et al. [5] for one of the few exceptions.

Similar bounds have recently been obtained by the authors for other special cases of the incidence problem [24, 25], using related but different approaches.

As an application, we prove that the number of incidences between m points and n lines in $\mathbb{R}^4$ contained in a quadratic hypersurface (which does not contain a hyperplane) is $O(m^{3/5}n^{3/5} + m + n)$.

All our bounds are significant improvements, under the restricted scenarios assumed in this work, over the standard incidence bounds in three dimensions, and shed, as we believe, new light on the structure of point-line incidences in three dimensions.

As is standard in the "modern" study of incidence geometry, the analysis is based on the *polynomial partitioning* technique (see [10, 11] for details), combined with a variety of tools from algebraic geometry. Due to lack of space, some details are missing in this version; they can be found in the full version [21].

2 Rich points determined by a two-dimensional family of lines

We first remark that, wherever needed in the analysis, we switch to the (projective 3-space over) the complex field, which simplifies it and lets us use numerous tools from algebraic geometry, available in this setting. The passage from the complex projective setup back to the real affine one is easy – the former is a generalization of the latter. The real affine setup is needed only for constructing a polynomial partitioning, which is meaningless over $\mathbb{C}$. Once we are, say, within the zero set $Z(f)$ of the partitioning polynomial f, we can switch to the complex projective setup, and reap the benefits just noted.

As already said, we parameterize lines in three dimensions by their *Plücker coordinates*, as follows (see, e.g., Griffiths and Harris [9, Section 1.5]). For a pair of distinct points $x, y \in \mathbb{P}^3$, given in projective coordinates as $x = (x_0, x_1, x_2, x_3)$ and $y = (y_0, y_1, y_2, y_3)$, let $\ell_{x,y}$ denote the (unique) line in $\mathbb{P}^3$ incident to both x and y. The Plücker coordinates of $\ell_{x,y}$ are given in projective coordinates in $\mathbb{P}^5$ as $(\pi_{0,1}, \pi_{0,2}, \pi_{0,3}, \pi_{2,3}, \pi_{3,1}, \pi_{1,2})$, where $\pi_{i,j} = x_i y_j - x_j y_i$. Under this parameterization, the set of lines in $\mathbb{P}^3$ corresponds bijectively to the set of points in $\mathbb{P}^5$ lying on the *Klein quadric* Q given by the quadratic equation

$$\pi_{0,1}\pi_{2,3} + \pi_{0,2}\pi_{3,1} + \pi_{0,3}\pi_{1,2} = 0 \tag{2}$$

(which is indeed always satisfied by the Plücker coordinates of a line).

Given a surface V in $\mathbb{P}^3$, the set of lines fully contained in V, represented by their Plücker coordinates in $\mathbb{P}^5$, is a subvariety of the Klein quadric Q, which is denoted by $F(V)$, and is called the *Fano variety* of V; see Harris [13, Lecture 6, page 63] and [13, Example 6.19].

Let H denote a plane in $\mathbb{R}^3$, and let H^* denote the 2-flat in the Plücker coordinates consisting of the points that represent the lines fully contained in H (see Rudnev [18] for why H^* is indeed a 2-flat and for more details).

For a set L of lines, we put $\mathcal{H}(L) = \{H^*_{\ell,\ell'} \mid \ell, \ell' \in L \text{ and } \ell, \ell' \text{ are coplanar}\}$, where for coplanar lines ℓ, ℓ', $H_{\ell,\ell'}$ is the (unique) 2-flat containing ℓ and ℓ'.

In this paper, we study incidences between a set of points $P \subset \mathbb{R}^3$, and a set of lines L in $\mathbb{R}^3$, whose Plücker images lie on some irreducible algebraic subvariety T of the Klein quadric Q, which is of constant degree, and which has dimension either 2 or 3.

In this section we restrict ourselves to the case where $\dim(T) = 2$. For a set L of lines, we say that the (two-dimensional) variety T is *non-degenerate* with respect to L if

(i) T is irreducible of constant degree,

(ii) T is not a 2-flat, and

(iii) the intersection of T with each 2-flat $H^* \in \mathcal{H}(L)$ consists of $O(1)$ Plücker points.

Note that condition (iii) is what one would expect to hold in a generic situation in a four-dimensional space. The simpler case where T is a 2-flat can be handled via the Szemerédi–Trotter theorem (Theorem 1) [28], but we will not consider this case in this work.

▶ **Theorem 6.**

(a) *Let L be a set of n lines in $\mathbb{R}^3$, such that, in Plücker space, L is a subset of some two-dimensional variety T that is non-degenerate with respect to L. Then the number of r-rich points determined by L is $|P_{\geqslant r}(L)| = O(n^{4/3+\varepsilon}/r^2)$, for any $\varepsilon > 0$ and $r \geqslant 3$.*

(b) *If the number of lines of L contained in any common regulus[2] is at most $n^{1/3}$ then the number of 2-rich points determined by L is $|P_{\geqslant 2}(L)| = O(n^{4/3+\varepsilon})$, for any $\varepsilon > 0$.*

Proof. First here is a high-level overview of the proof. After a pruning step, we may assume that the set γ_ℓ, for a line $\ell \in L$, of the lines coplanar with ℓ and lying in T, is a one-dimensional curve in T. An r-rich point generates $\Omega(r^2)$ incidences between the Plücker points of the lines of L and the curves γ_ℓ, so it suffices to bound the number of such incidences. There are two kinds of curves, those that represent the lines in one ruling of some regulus, and those that do not. For r-rich points, with $r \geqslant 3$, only the latter kind of curves matter, and a suitable application of Theorem 3 allows us to obtain an upper bound for the number of such incidences. For 2-rich points (part (b) of the theorem), the regulus-curves also play a part, and the analysis is complicated because these curves do not have to be distinct. Still, the assumptions in (b) allow us to handle this case and get the desired bound.

To simplify the presentation, we use the same notation for a line in 3-space and for its Plücker point in Q (we will deviate from this convention in Section 5). The following notation will also be useful later on in the paper. For each line $\ell \in Q$, define the variety S_ℓ to be

$$S_\ell = \{\ell' \in Q \mid \text{ and } \ell, \ell' \text{ are coplanar}\}.$$

If the Plücker coordinates of ℓ are $(\pi_{0,1}, \pi_{0,2}, \pi_{0,3}, \pi_{2,3}, \pi_{3,1}, \pi_{1,2})$, then

$$S_\ell = \{(\pi'_{0,1}, \pi'_{0,2}, \pi'_{0,3}, \pi'_{2,3}, \pi'_{3,1}, \pi'_{1,2}) \in Q \mid \\ \pi_{0,1}\pi'_{2,3} + \pi_{0,2}\pi'_{3,1} + \pi_{0,3}\pi'_{1,2} + \pi'_{0,1}\pi_{2,3} + \pi'_{0,2}\pi_{3,1} + \pi'_{0,3}\pi_{1,2} = 0\}.$$

In particular, Equation (2) implies that $\ell \in S_\ell$. We see that, for every line ℓ, the variety S_ℓ is the intersection of Q with a hyperplane, so it is a three-dimensional quadratic surface contained in Q, and we clearly have $\ell \in S_\ell$. We say that a line ℓ is *exceptional* with respect to T if $T \subset S_\ell$. We say that a point $p \in \mathbb{R}^3$ is *exceptional* with respect to T if the set of lines incident to p in 3-space, which we denote by S_p and which is known to be a 2-plane (see the proof below), is equal to T. Clearly, since T is non-degenerate, there are no exceptional points with respect to T (see the full version [21] for an additional discussion).

[2] This assumption is needed only for bounding the number of 2-rich points.

▶ **Lemma 7.** *There are at most two exceptional lines with respect to T.*

Proof. Assume to the contrary that there are three exceptional lines. Assume first that two of these lines are coplanar, and denote them by ℓ_1 and ℓ_2. Then $T \subseteq S_{\ell_1} \cap S_{\ell_2}$, i.e., T is contained in the set of lines intersecting both ℓ_1 and ℓ_2. If ℓ_1 and ℓ_2 do not intersect one another, then $S_{\ell_1} \cap S_{\ell_2} = H^*_{\ell_1,\ell_2}$. Otherwise, letting $p = \ell_1 \cap \ell_2$, we have $S_{\ell_1} \cap S_{\ell_2} = H^*_{\ell_1,\ell_2} \cup S_p$, i.e., it is a union of two 2-flats. In both cases, T is a 2-flat, contradicting our assumption.

We may thus assume there are (at least) three lines ℓ_1, ℓ_2 and ℓ_3 that are pairwise skew, such that $T \subseteq S_{\ell_1} \cap S_{\ell_2} \cap S_{\ell_3}$. As is well known (see, e.g., [7, Theorem 16.4] and [23, Lemma 2.2]), the Plücker points of lines that intersect $r \geqslant 3$ pairwise-skew lines $\ell_1, \ldots, \ell_r$ belong to one ruling of a regulus, and the Plücker points of $\ell_1, \ldots, \ell_r$ belong to the other ruling of this regulus. That is, $S_{\ell_1} \cap S_{\ell_2} \cap S_{\ell_3}$ is one ruling of the regulus generated by the lines intersecting ℓ_1, ℓ_2 and ℓ_3, which is a quadratic curve in the Plücker space, contradicting the fact that T is two-dimensional. ◀

We prune away, as we may, the (at most) two exceptional lines, thereby losing at most $2(n-1) < 2n$ 2-rich points.

For each of the (remaining) lines $\ell \in L$, the intersection $S_\ell \cap T$ is a curve contained in T (possibly also containing a discrete finite subset), which we denote by γ_ℓ. Define

$$\mathcal{C} = \{\gamma_\ell \mid \ell \text{ is not exceptional}\}. \tag{3}$$

We have the following simple observation, whose trivial proof is omitted.

▶ **Lemma 8.** *Let $p \in P$ be an r-rich point, with $r \geqslant 2$, and denote the lines of L incident to p as $\ell_1, \ldots, \ell_s$, for some $s \geqslant r$. Then, for each pair of indices $1 \leqslant i \neq j \leqslant s$, ℓ_i, viewed as a point in Q, is incident to γ_{ℓ_j}, and for every such incidence there is at most one point $p \in P$ that induces it, in the sense stated above.*

The lemma asserts that each r-rich point contributes at least $r(r-1)$ incidences between the lines of L (as points in Q) and the curves γ_ℓ of $\mathcal{C}$ (as curves in Q). Hence, to bound the number of r-rich points, it suffices to bound the number of incidences between the lines in L and the curves of $\mathcal{C}$ (and then divide the bound by $r(r-1)$). For any curve γ_ℓ, its corresponding discrete subset of $O(1)$ points contributes only $O(1)$ incidences, for a total of $O(n)$ incidences. We may therefore ignore all these discrete subsets.

The notion of *dimensionality* for families of curves (see the definition preceding Theorem 3) easily extends in a natural way to collections $\mathcal{C}$ of higher-dimensional algebraic varieties.

▶ **Lemma 9.** *The family $\mathcal{C}$ defined in (3) is two-dimensional.*

Proof. Each curve γ_ℓ of $\mathcal{C}$ can be parameterized by the parameters of the corresponding line ℓ, and these lines lie in the two-dimensional variety T in the Plücker parametric space. ◀

Denote by L^* the set of points in the Klein quadric Q that represent the lines of L. When analyzing incidences between the points of L^* and the curves γ_j, as in Lemma 8, some care has to be exercised, to handle situations in which many of the curves γ_ℓ share a common irreducible component (or even coincide).

($\geqslant 3$)-rich points. Assume that $\ell_1, \ldots, \ell_\xi \in L$ are such that $\gamma_{\ell_1}, \ldots, \gamma_{\ell_\xi}$ all share a common curve, for some $\xi \geqslant 3$. If some pair of lines ℓ_i, ℓ_j are coplanar, we write $H_{i,j}$ for the (unique) plane H_{ℓ_i,ℓ_j} containing both ℓ_i and ℓ_j. As in the proof of Lemma 7, (i) if ℓ_i and ℓ_j are parallel then $S_{\ell_i} \cap S_{\ell_j} = H^*_{i,j}$, and (ii) if ℓ_i and ℓ_j intersect in a point p then $S_{\ell_i} \cap S_{\ell_j} = H^*_{i,j} \cup S_p$, so $S_{\ell_i} \cap S_{\ell_j}$ is either a 2-flat or the union of two 2-flats. Therefore,

$$\gamma_{\ell_i} \cap \gamma_{\ell_j} = S_{\ell_i} \cap S_{\ell_j} \cap T = H^*_{i,j} \cap T \quad \text{or} \quad (H^*_{i,j} \cup S_p) \cap T,$$

and the right hand sides of these equations are the intersection of one or two 2-flats with T. Since T is assumed to be non-degenerate, it follows that $\gamma_{\ell_i} \cap \gamma_{\ell_j}$ is a finite set of points, and thus γ_{ℓ_i} and γ_{ℓ_j} cannot intersect in a common curve. We can thus assume that $\ell_1, \ldots, \ell_\xi$ are pairwise skew (and that $\gamma_{\ell_1}, \ldots, \gamma_{\ell_\xi}$ intersect in a common curve).

▶ **Lemma 10.** *Assume that the arc (in Plücker space) $\gamma := \bigcap_{i=1}^{\xi} \gamma_{\ell_i}$ is nonempty (and is not a finite set), where $\ell_1, \ldots, \ell_\xi$ are ξ pairwise-skew lines, $\xi \geqslant 3$. Then γ parameterizes one ruling of a regulus, and, for each line $\ell \in T$ whose Plücker point is in γ, ℓ intersects $\ell_1, \ldots, \ell_\xi$, so its Plücker point lies in the curve that represents the other ruling of the same regulus.*

Proof. The proof is similar to that of Lemma 7. The intersection $\bigcap_{i=1}^{\xi} S_{\ell_i}$ consists of the Plücker points of the lines that intersect the $\xi \geqslant 3$ pairwise-skew lines $\ell_1, \ldots, \ell_\xi$. Thus, as already noted (see [7]), all these lines belong to one ruling of a regulus, and the Plücker points of $\ell_1, \ldots, \ell_\xi$ belong to the other ruling of this regulus. Therefore, γ parameterizes one ruling of a regulus, and $\ell_1, \ldots, \ell_\xi$ belong to the other ruling of this regulus, as asserted. ◀

Partition the set of irreducible components of the curves γ_ℓ, over all lines $\ell \in L$ that are not exceptional, into two subsets $\mathcal{C}_0$ and $\mathcal{C}_1$, where $\mathcal{C}_0$ (resp., $\mathcal{C}_1$) contains all the components that do not (resp., do) parameterize one ruling of some regulus. Since $\deg(\gamma_\ell) \leqslant \deg(T) = O(1)$, for each $\ell \in L$, it follows that $|\mathcal{C}_0| = |\mathcal{C}_1| = O(n)$. We partition the set of incidences into incidences between the Plücker points of the lines in L and the curves in $\mathcal{C}_0$, and incidences with the curves in $\mathcal{C}_1$. We remind the reader that at this stage we are only concerned with incidences induced by a concurrence of $r \geqslant 3$ lines of L at some (r-rich) point p.

By Lemma 8, any r-rich point p, for $r \geqslant 3$, corresponds to incidences between the points in Plücker space that represent lines ℓ of L that are incident to p and the (at least three) curves $\gamma_{\ell'}$ that are associated with these lines, and any such incidence can arise for at most one point p. One possibility is that the Plücker point of a line ℓ (incident to p), is incident to a common component of at least three of these curves, call them $\gamma_{\ell_1^p}, \gamma_{\ell_2^p}, \gamma_{\ell_3^p}$. However, the analysis preceding Lemma 10 implies that ℓ_1^p, ℓ_2^p, ℓ_3^p are pairwise skew, which is impossible as they are all incident to p. Hence an incidence between a line and a common component of at least three curves γ_{ℓ_i} does not generate any r-rich points, for $r \geqslant 3$, and, by construction, curves in $\mathcal{C}_1$ also do not generate any r-rich points, for $r \geqslant 3$. We may therefore assume that every curve in $\mathcal{C}_0$ is an irreducible component of at most two curves in $\mathcal{C}$.

Summarizing, the number of r-rich points, with $r \geqslant 3$, is proportional to the number of incidences between the Plücker points of the lines in L and the distinct curves in $\mathcal{C}_0$, divided by $\binom{r}{2}$, and there is no contribution by the curves in $\mathcal{C}_1$.

2-rich points. The situation is different for 2-rich points, which may arise also as incidences between the Plücker points of lines in L and curves in $\mathcal{C}_1$. Handling them requires more care, and is done as follows. A *proper* 2-rich point p, namely a point that is incident to precisely two lines ℓ_p and ℓ'_p of L, corresponds to an incidence between the Plücker point of ℓ_p and the curve $\gamma_{\ell'_p}$ (and also between the Plücker point of ℓ'_p and the curve γ_{ℓ_p}). We count this incidence at most $\deg(\gamma_{\ell'}) = O(1)$ times, once for each irreducible component of the curve

$\gamma_{\ell'_p}$. It therefore suffices to count incidences between the Plücker points of the lines in L and the curves in $\mathcal{C}_0$ (as we have just argued, this is relevant only for curves of multiplicity at most 2) and in $\mathcal{C}_1$ (which may have an arbitrary multiplicity).

We now combine the arguments in the two subcases in the final phase of the analysis. Consider first incidences with curves of $\mathcal{C}_0$. By projecting T onto some generic plane, the number of incidences between the n points representing the lines of L and the curves of $\mathcal{C}_0$ is the same as the number of incidences between the projected points and the projected curves. Since $\mathcal{C}$ is a two-dimensional family of curves (Lemma 9), so is $\mathcal{C}_0$. It therefore follows, by Theorem 3, that the number of these incidences is $O(n^{4/3+\varepsilon})$, for any $\varepsilon > 0$. As argued above, this gives us the bound $O(n^{4/3+\varepsilon}/r^2)$ on the number of r-rich points, for $r \geqslant 3$, thereby establishing part (a) of Theorem 6.

This also gives us the bound $O(n^{4/3+\varepsilon})$ for the number of 2-rich points that correspond to incidences formed with the curves of $\mathcal{C}_0$. For the remaining 2-rich points, which correspond to incidences between lines in L (points of L^*) with curves of $\mathcal{C}_1$ (which may appear with arbitrarily large multiplicity), we recall that each of the curves in $\mathcal{C}_1$ represents one ruling of some regulus, and that we have assumed that no regulus contains more than $n^{1/3}$ lines of L. Hence the each curve in $\mathcal{C}_1$ is incident to at most $n^{1/3}$ lines in L (points in L^*), which implies that the number of incidences with these curves, counted with multiplicity, is at most $O(n^{4/3})$. Hence part (b) of the theorem also follows, and the proof is thus completed. ◀

3 Application: Distinct distances between points on an algebraic curve in the plane

Let P be a set of n points on an irreducible algebraic curve γ of constant degree in the plane, which is not a line or a circle. We derive a lower bound on the number of distinct distances between the points of P; we only sketch our approach, with details in the full version [21]. To derive the bound, we apply the Elekes-Sharir-Guth-Katz framework [6, 11], and define a set L of $n(n-1)$ lines in the parametric 3-space of rotations (rigid motions) in the plane, as $L = \{h_{a,b} \mid a \neq b \in P\}$, where $h_{a,b}$ is the locus of all rotations that map a to b. Then L is contained in the two-dimensional family of lines $\mathcal{C} = \{h_{x,y} \mid x, y \in \gamma\}$. We show that $\mathcal{C}$ is irreducible and is not a 2-flat, and, after pruning away some lines of L, the number of remaining lines in L that are contained in a common plane or regulus in 3-space is $O(1)$, and thus $\mathcal{C}$ is non-degenerate with respect to L. We can then apply the machinery of the previous section to L and $\mathcal{C}$, and derive our bound.

In more detail, let Δ denote the number of distinct distances determined by P. We count the number of quadruples $\{(a, b, a', b') \in P^4 \mid |ab| = |a'b'|\}$ in two different ways. First, let N_k (resp., $N_{\geqslant k}$) denote the number of rotations of multiplicity exactly (resp., at least) k; that is, rotations that map exactly (resp., at least) k points of P to k other points of P. By construction, a rotation of multiplicity at least k is mapped to a k-rich point with respect to the lines of L. Then Theorem 6 implies

$$N_{\geqslant k} = O((n^2)^{4/3+\varepsilon/2}/k^2) = O(n^{8/3+\varepsilon}/k^2), \tag{4}$$

provided that the number of lines in L contained in a common plane or regulus is $O(|L|^{1/3})$, a property that we establish. In fact, we show that no plane or regulus contains more than a constant number of lines of L, except for at most $O(1)$ special planes, whose effect on the asserted bound is negligible, and which we ignore by removing all the lines contained in these planes. Hence, arguing as in [6, 11] and using (4), the number of quadruples is at most

$$\sum_{k=2}^{n}\binom{k}{2}N_k \leqslant \sum_{k=2}^{n} kN_{\geqslant k} = O\left(\sum_{k=2}^{n}\frac{n^{8/3+\varepsilon}}{k}\right) = O(n^{8/3+\varepsilon}\log n) = O(n^{8/3+2\varepsilon}). \tag{5}$$

On the other hand, by Elekes's analysis, which is based on the Cauchy-Schwarz inequality (see, e.g., Guth and Katz [6, 11]), the number of quadruples is also $\Omega(n^4/\Delta)$, implying that the number of distinct distances satisfies $\Delta = \Omega(n^{4/3-2\varepsilon})$. This result was obtained earlier in [16], without the ε-loss in the exponent, but the proof here is much simpler, and we hope that it will find similar applications of this kind.

4 Incidences between points and lines in a two-dimensional family of lines

The main result of this section is the following theorem.

▶ **Theorem 11.** *Let P be a set of m points in $\mathbb{R}^3$, and let L be a set of n lines in $\mathbb{R}^3$, such that, in Plücker coordinates, L is contained in some two-dimensional, non-planar, irreducible variety T of constant degree, as in Section 2. Then $I(P,L) = O(m+n)$.*

Proof. We only provide a sketch here; full details are given in the full version [21]. As observed above, for a point $p \in \mathbb{C}^3$, the set of lines S_p that are incident to p form a 2-flat in the parametric Plücker space, which is contained in Q. We assume that for every $p \in \mathbb{C}^3$, $T \neq S_p$; otherwise, all the lines in T would be incident to p, so the number of incidences would be $O(m+n)$, as asserted.

As T is two-dimensional, Bézout's theorem [8] implies that for every $p \in \mathbb{C}^3$, the intersection $S_p \cap T$ is a union of a constant number of curves of constant degree and a discrete set of a constant number of points.

Put $V := \{p \in \mathbb{C}^3 \mid S_p \cap T \text{ is a curve}\}$. Similar to [22, Theorem 2.16], one can define a polynomial of constant degree, via multivariate resultants, whose vanishing at a point p is equivalent to $S_p \cap T$ being one-dimensional. Hence, V is a complex algebraic variety, and, as T is of constant degree, so is V. We may assume that V is irreducible.

We argue that if $V = \mathbb{C}^3$ then T has to be three-dimensional, so we have $V \neq \mathbb{C}^3$. We then argue that V cannot be two-dimensional, using a somewhat involved argument, whose details are given in [21]. Hence, V must be one-dimensional. By definition of V, every point $p \in P \setminus V$ is incident to at most $O(1)$ lines of L, for a total of $O(m)$ incidences. Thus, we may assume that all the points of P are contained in the curve V. For each line $\ell \in L$, if ℓ is not contained in V it contributes at most $O(\deg V) = O(1)$ incidences with P. Thus, we get a total of $O(n)$ incidences, except for at most $O(\deg V) = O(1)$ lines that are contained in V, for a total of $O(m)$ additional incidences. This completes the proof of the theorem. ◀

The following corollary is an immediate consequence of the theorem.

▶ **Corollary 12.** *Let T be a two-dimensional, non-planar, irreducible subvariety, of constant degree, of the Klein quadric Q. Then, there exists a constant $r_0 = r_0(\deg(T))$ so that, if L is a set of n lines in $\mathbb{R}^3$ whose Plücker images are points in T then, for $r \geqslant r_0$, the set $P_{\geqslant r}(L)$ of r-rich points determined by L satisfies $|P_{\geqslant r}(L)| = O(n/r)$.*

5 Incidences between points and lines in a three-dimensional family of lines

In this section we prove the following main result.

▶ **Theorem 13.** *The number of incidences between m points in $\mathbb{R}^3$ and n lines in $\mathbb{R}^3$ whose Plücker images are contained in an irreducible nonlinear constant-degree three-dimensional variety T is*

$$O\left(m^{3/5}n^{3/5} + (m^{11/15}n^{2/5} + m^{1/3}n^{2/3})s^{1/3} + m + n\right),$$

provided that no plane contains more than s of the points. If $s = O(\min\{n^{3/5}/m^{2/5}, m^{1/2}\})$, the bound becomes $O(m^{3/5}n^{3/5} + m + n)$.

Proof. Recal that S_p is the 2-flat in Plücker space that consists of all lines passing through a point $p \in \mathbb{R}^3$, and define

$$W := \{p \in \mathbb{C}^3 \mid S_p \cap T \text{ is two-dimensional}\}.$$

▶ **Lemma 14.** *W is an algebraic variety of dimension at most 2 and of constant degree.*

Proof. Since S_p is a 2-flat, $S_p \cap T$ is two-dimensional if and only if $S_p \subset T$. Similarly to the Fano variety of lines, the Grassmannian manifold of 2-flats contained in a constant-degree variety is an algebraic variety [9] of constant degree, so W is algebraic of constant degree. To bound the dimension of W, we repeat the proof of [23, Theorem 2.3(a)], which proceeds by counting the dimensions of the fibers that arise in the problem. Here we omit the details and give the high-level idea. Assume to the contrary that W is three-dimensional, i.e., $W = \mathbb{C}^3$, so for every point $p \in \mathbb{C}^3$, the 2-flat S_p is contained in T. Omitting details, we note that each $p \in \mathbb{C}^3$ contributes a two-dimensional set ($\dim(S_p) = 2$), but then every line is counted by the infinitely many points incident to it. A standard dimension counting argument then implies that $\dim(F(T)) \geqslant 4$, where $F(T)$ is the Fano variety of lines contained in T. By [22, Theorem 3.11], this implies that T has to be a 3-flat, contradicting our assumption. ◀

We first treat incidences with points $p \in P \cap W$. We decompose W into its $O(1)$ irreducible components, and treat each component separately. If a component W_0 of W is not a 2-flat then, by [23, Corollary 1.4], the number of incidences between points contained in W_0 and lines in L is $O(m+n)$. If W_0 is a 2-flat, we invoke the Szemerédi-Trotter bound in Theorem 1, and get the bound $O(s^{2/3}n^{2/3} + s + n)$, using our assumption that no 2-flat contains more than s points of P. This in turn can be upper bounded by $O(s^{1/3}m^{1/3}n^{2/3} + m + n)$, which is subsumed by the bound asserted by the theorem.

Next, we treat incidences involving points in $\mathbb{R}^3 \setminus W$, i.e., points that are incident to a one-dimensional family of lines in T. In this case we use duality, replacing each point p in $\mathbb{R}^3$ with the one-dimensional curve γ_p of lines incident to p and contained in T, in Plücker space. This yields a family of m constant-degree curves that is a family of pseudo-lines. (Two such curves γ_p and γ_q intersect in at most one point, corresponding to the (unique) line connecting p and q, if it lies in T.) We replace each of the n lines in L by its Plücker image, and obtain an incidence problem between n points and m pseudo-lines within the variety T, a three-dimensional subset of the four-dimensional Klein quadric Q. Using a generic projection of T onto $\mathbb{R}^3$ (in which all projected points are distinct and no pair of

projected curves overlap), the analysis then proceeds by invoking Zahl [29, Lemma 4.1], which extends the Guth–Katz incidence bound from incidences with lines to incidences with pseudo-lines. Specifically, Zahl shows that the number of incidences between n points and m pseudo-lines in $\mathbb{R}^3$, assuming that these pseudo-lines are constant-degree algebraic curves, is $O(n^{1/2}m^{3/4}+n^{2/3}m^{1/3}\xi^{1/3}+m+n)$, where ξ is an upper bound on the number of pseudo-lines that are contained in any common two-dimensional surface contained in T that is infinitely ruled by curves from the infinite family from which our pseudo-lines are taken.

As argued in Guth and Zahl [12], any such surface must be of degree at most $100E^2$, where E is the degree of the pseudo-lines γ_p, so it is sometimes convenient, especially when no simple characterization of such infinitely ruled surfaces is known, to impose the stronger assumption that no surface of degree at most $100E^2$ contains more than ξ pseudo-lines. In general, this assumption is too restrictive, and difficult to verify. One of the main technical contribution of the analysis in this section is to exploit the dual nature of the present setup, and replace this assumption by the simpler and more natural assumption that, in the original "primal" 3-space, *no plane contains more than s points of P*, allowing us to replace ξ by s, and obtain the incidence bound

$$I(P,L)=O(n^{1/2}m^{3/4}+n^{2/3}m^{1/3}s^{1/3}+m+n).$$

Since $n^{1/2}m^{3/4}\leqslant m^{3/2}$ when $n\leqslant m^{3/2}$, and $n^{1/2}m^{3/4}\leqslant n$ otherwise, we get the following bootstrapping bound

$$I(P,L)=O(m^{3/2}+n^{2/3}m^{1/3}s^{1/3}+n). \tag{6}$$

The analysis then proceeds by "starting over" in primal space, i.e., by constructing a partitioning polynomial g of degree $O(D)$, for a suitable value of D, to be fixed shortly, using the techniques in [10, 11], so that each connected component (cell) τ of $R^3\setminus Z(g)$ contains at most m/D^3 points of P and is crossed by at most n/D^2 lines of L (but any number of points and lines can be contained in the zero set $Z(g)$).

Incidences within the cells. We first bound the number of incidences within the partition cells. We apply the bootstrapping bound in (6) to each cell τ and sum the bound over all components, to obtain the bound

$$\begin{aligned}O\left(D^3((m/D^3)^{3/2}+(n/D^2)^{2/3}(m/D^3)^{1/3}s^{1/3}+n/D^2)\right)\\ =O\left(\frac{m^{3/2}}{D^{3/2}}+n^{2/3}m^{1/3}D^{2/3}s^{1/3}+nD\right).\end{aligned}$$

To balance the first and last terms, we choose $D=m^{3/5}/n^{2/5}$. For this to make sense, we require that $1\leqslant D\leqslant\min\{m^{1/3},n^{1/2}\}$, or, equivalently, that $n\leqslant m^{3/2}$ and $m\leqslant n^{3/2}$. When the first inequality does not hold, we do not use any partitioning and just apply (6) to obtain the bound $I(P,L)=O(n^{2/3}m^{1/3}s^{1/3}+n)$. When the second inequality does not hold, we choose $D=an^{1/2}$, for a suitable constant a, which satisfies the inequalities. In fact, we can construct a polynomial g of this degree so that all the lines of L are fully contained in $Z(g)$ (see, e.g., [14]), and we may therefore assume that all the points of P are also contained in $Z(g)$, as the other points contribute no incidences. That is, in this case there are no incidences within the cells.

In the middle range, our choice of D yields the bound $O(m^{3/5}n^{3/5}+m^{11/15}n^{2/5}s^{1/3})$. Combining all the bounds, the number of incidences within the partition cells is

$$O\left(m^{3/5}n^{3/5}+(m^{11/15}n^{2/5}+m^{1/3}n^{2/3})s^{1/3}+m+n\right). \tag{7}$$

Incidences on the zero set. Consider next incidences involving points that lie on $Z(g)$. A line ℓ that is not fully contained in $Z(g)$ crosses it in at most $O(D)$ points, for an overall $O(nD)$ bound, which is subsumed by the bound (7) for incidences within the cells. It therefore remains to bound the number of incidences between the points of P on $Z(g)$ and the lines that are fully contained in $Z(g)$.

We handle each irreducible component of $Z(g)$ separately. For non-planar components, Theorem 5, combined with Hölder's inequality (for summing up the bounds over the irreducible components) implies that the number of incidences between points and lines contained in $Z(g)$, but not in any planar component of $Z(g)$, is

$$O(m^{1/2}n^{1/2}D^{1/2} + m^{1/3}D^{4/3}s^{1/3} + m + n).$$

In the middle range $n^{2/3} \leqslant m \leqslant n^{3/2}$, the choice of $D = m^{3/5}/n^{2/5}$ is easily seen to yield the desired bound $O(m^{3/5}n^{3/5} + m + n)$. The case $m < n^{2/3}$ has already been handled, by a single application of (6), which yields the bound $O(n^{2/3}m^{1/3}s^{1/3} + n)$. When $m > n^{3/2}$, the choice of $D = an^{1/2}$, as made above, yields the bound $O(n^{2/3}m^{1/3}s^{1/3} + m)$.

For the planar components, we use the standard technique of assigning each point and line to the first planar component that contains it (according to some arbitrary enumeration of the components). The number of incidences between points and lines assigned to different components is $O(nD) = O(m^{3/5}n^{3/5} + m + n)$ (the right-hand side does indeed bound the left-hand side for each of the sub-ranges). For incidences between points and lines assigned to the same planar component, we apply the Szemerédi-Trotter bound (Theorem 1) to each component and sum the resulting bounds over the components. The assumption that each plane contains at most s points, combined with Hölder's inequality, yields the bound $O(m^{1/3}n^{2/3}s^{1/3} + m + n)$.

That is, the number of incidences with points on $Z(g)$ is bounded by

$$O\left(m^{3/5}n^{3/5} + m^{1/3}n^{2/3}s^{1/3} + m + n\right). \tag{8}$$

Combining with the bound (7) for incidences within the cells, we get the overall bound

$$I(P, L) = O\left(m^{3/5}n^{3/5} + (m^{11/15}n^{2/5} + m^{1/3}n^{2/3})s^{1/3} + m + n\right),$$

thereby completing the proof of the theorem. ◀

▶ Remark. An interesting challenge in incidence geometry is to sharpen the Guth-Katz bound [11] when the number of lines in any common plane is at most some constant. When the lines in L are contained, as points in Plücker space, in an irreducible nonlinear constant-degree three-dimensional variety T then, while we cannot deduce that the number of lines contained in a common plane is constant, we can nevertheless show: For any plane $\Pi \subset \mathbb{C}^3$, $T \cap \Pi^*$ (recall that Π^* is the 2-flat dual to Π, consisting of all the points dual to lines that are contained in Π) is a constant-degree curve, and thus, except for $O(1)$ points, every point in Π is incident to $O(1)$ lines in T, implying that the number of incidences in a common plane is $O(m_\Pi + n_\Pi)$, where m_Π (n_Π) is the number of points (lines) contained in Π. Such a linear bound on the number of incidences within a plane is a key property for deriving improved incidence bounds, as demonstrated in this work. For Theorem 13, we also added the condition that $m_\Pi \leqslant s$, for every plane Π, to further improve the bound.

6 Application: Incidences between points and lines on a quadric in four dimensions

Solomon and Zhang [26] give a configuration of m points and n lines in a quadratic hypersurface in $\mathbb{R}^4$, having $\Omega(m^{2/3}n^{1/2}+m+n)$ incidences. The following theorem follows as a corollary from the previous section.

▶ **Theorem 15.** *Let P be a set of m points and L a set of n lines contained in a quadratic hypersurface $S \subset \mathcal{C}^4$ such that no 2-flat contains more than $s = O(n^{3/5}/m^{2/5})$ of the points of P. Then $I(P,L) = O(m^{3/5}n^{3/5}+m+n)$.*

▶ Remark. When $m = O(n^{3/2})$, the lower bound $\Omega(m^{2/3}n^{1/2}+m+n)$ obtained in [26] is (asymptotically) smaller than the upper bound $O(m^{3/5}n^{3/5}+m+n)$ asserted in Theorem 15. Closing this gap remains a challenging open problem.

References

1 P. Agarwal, E. Nevo, J. Pach, R. Pinchasi, M. Sharir, and S. Smorodinsky. Lenses in arrangements of pseudocircles and their applications. *J. ACM*, 51:139–186, 2004.
2 B. Aronov and M. Sharir. Cutting circles into pseudo-segments and improved bounds for incidences. *Discrete Comput. Geom.*, 28:475–490, 2002.
3 J. Cardinal, M. Payne, and N. Solomon. Ramsey-type theorems for lines in 3-space. *Discrete Math. Theoret. Comput. Sci.*, 18, 2016.
4 K. Clarkson, H. Edelsbrunner, L. Guibas, M. Sharir, and E. Welzl. Combinatorial complexity bounds for arrangements of curves and spheres. *Discrete Comput. Geom.*, 5:99–160, 1990.
5 G. Elekes, H. Kaplan, and M. Sharir. On lines, joints, and incidences in three dimensions. *J. Combinat. Theory, Ser. A*, 118:962–977, 2011. Also in `arXiv:0905.1583`.
6 Gy. Elekes and M. Sharir. Incidences in three dimensions and distinct distances in the plane. *Combinat. Probab. Comput.*, 20:571–608, 2011. Also in `arXiv:1005.0982`.
7 D. Fuchs and S. Tabachnikov. *Mathematical Omnibus: Thirty Lectures on Classic Mathematics*. Amer. Math. Soc. Press, Providence, RI, 2007.
8 W. Fulton. *Introduction to Intersection Theory in Algebraic Geometry*. Expository Lectures from the CBMS Regional Conference Held at George Mason University, June 27–July 1, 1983, Vol. 54. AMS Bookstore, 1984.
9 P. Griffiths and J. Harris. *Principles of Algebraic Geometry*, volume 52. John Wiley & Sons, New York, 2011.
10 L. Guth. Polynomial partitioning for a set of varieties. *Math. Proc. Cambridge Phil. Soc.*, 159:459–469, 2015. Also in `arXiv:1410.8871`.
11 L. Guth and N.H̃. Katz. On the ErdHos distinct distances problem in the plane. *Annals Math.*, 181:155–190, 2015. Also in `arXiv:1011.4105`.
12 L. Guth and J. Zahl. Algebraic curves, rich points, and doubly-ruled surfaces. *Amer. J. Math.*, 140:1187–1229, 2018. Also in `arXiv:1503.02173`.
13 J. Harris. *Algebraic Geometry: A First Course*, volume 133. Springer-Verlag, New York, 1992.
14 H. Kaplan, M. Sharir, and E. Shustin. On lines and joints. *Discrete Comput. Geom.*, 44:838–843, 2010. Also in `arXiv:0906.0558`.
15 A. Marcus and G. Tardos. Intersection reverse sequences and geometric applications. *J. Combinat. Theory, Ser. A*, 113:675–691, 2006.
16 J. Pach and F. de Zeeuw. Distinct distances on algebraic curves in the plane. *Combinat. Probab. Comput.*, 26:99–117, 2017.
17 J. Pach and M. Sharir. On the number of incidences between points and curves. *Combinat. Probab. Comput.*, 7:121–127, 1998.
18 M. Rudnev. On the number of incidences between points and planes in three dimensions. *Combinatorica*, 38:219–254, 2018.

19 M. Sharir, A. Sheffer, and N. Solomon. Incidences with curves in R^d. *Electronic J. Combinat.*, 23(4), 2016. Also in *Proc. European Sympos. Algorithms* (2015), Springer LNCS 9294, pp. 977–988, and in `arXiv:1512.08267`.

20 M. Sharir, A. Sheffer, and J. Zahl. Improved bounds for incidences between points and circles. *Combinat. Probab. Comput.*, 24:490–520, 2015. Also in `arXiv:1208.0053`.

21 M. Sharir and N. Solomon. On rich points and incidences with restricted sets of lines in 3-space. `arXiv:2012.11913`.

22 M. Sharir and N. Solomon. Incidences between points and lines in R^4. *Discrete Comput. Geom.*, 57(3):702–756, 2017. Also in *Proc. 56th IEEE Sympos. Foundations of Computer Science* 2015, 1378–1394, and in `arXiv:1411.0777`.

23 M. Sharir and N. Solomon. Incidences between points and lines on two- and three- dimensional varieties. *Discrete Comput. Geom.*, 59:88–130, 2018. Also in `arXiv:1609.09026`.

24 M. Sharir, N. Solomon, and O. Zlydenko. Incidences with curves with almost two degrees of freedom. `arXiv:2003.02190`. Also in *Proc. 36th Sympos. on Computational Geometry* (2020), 66:1–66:14.

25 M. Sharir and J. Zahl. Cutting algebraic curves into pseudo-segments and applications. *J. Combinat. Theory Ser. A*, 150:1–35, 2017.

26 N. Solomon and R. Zhang. Highly incidental patterns on a quadratic hypersurface in R^4. *Discrete Math.*, 340:585–590, 2017.

27 L. Székely. Crossing numbers and hard ErdHos problems in discrete geometry. *Combinat. Probab. Comput.*, 6:353–358, 1997.

28 E. Szemerédi and W.T. Trotter. Extremal problems in discrete geometry. *Combinatorica*, 3:381–392, 1983.

29 J. Zahl. Breaking the 3/2 barrier for unit distances in three dimensions, 2017. `arXiv:1706.05118`.

Sketching Persistence Diagrams

Donald R. Sheehy
North Carolina State University, Raleigh, NC, USA

Siddharth Sheth
North Carolina State University, Raleigh, NC, USA

Abstract

Given a persistence diagram with n points, we give an algorithm that produces a sequence of n persistence diagrams converging in bottleneck distance to the input diagram, the ith of which has i distinct (weighted) points and is a 2-approximation to the closest persistence diagram with that many distinct points. For each approximation, we precompute the optimal matching between the ith and the $(i+1)$st. Perhaps surprisingly, the entire sequence of diagrams as well as the sequence of matchings can be represented in $O(n)$ space. The main approach is to use a variation of the greedy permutation of the persistence diagram to give good Hausdorff approximations and assign weights to these subsets. We give a new algorithm to efficiently compute this permutation, despite the high implicit dimension of points in a persistence diagram due to the effect of the diagonal. The sketches are also structured to permit fast (linear time) approximations to the Hausdorff distance between diagrams – a lower bound on the bottleneck distance. For approximating the bottleneck distance, sketches can also be used to compute a linear-size neighborhood graph directly, obviating the need for geometric data structures used in state-of-the-art methods for bottleneck computation.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Bottleneck Distance, Persistent Homology, Approximate Persistence Diagrams

Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.57

Related Version *Full Version*: https://arxiv.org/abs/2012.01967

Funding This research was supported by the NSF under grant CCF-2017980.

1 Introduction

Persistent homology (PH) is a topological invariant with a built-in metric. Thus, qualitative shape information (topology) becomes quantitative (distances). This is why PH is so useful as a meta-analysis tool; it can map an entire data set to a single point in a metric space, i.e., a *persistence diagram* (PD). The complexity of computing the distance between PDs is determined by the complexity of the PDs themselves, which are multisets of pairs of numbers. The exact distance is computed by finding the minimum bottleneck matching between the sets. Naturally, smaller diagrams lead to faster computation.

In this paper, we will explore methods for sketching PDs, producing much smaller diagrams while maintaining some guaranteed proximity to the original PD. Given a PD D with n distinct (nondiagonal) points, we will produce a sequence of PDs $D_0, \ldots, D_n$ where each D_i has i distinct points. Let ε_i be the bottleneck distance $\mathrm{d}_B(D, D_i)$ for all i. The sequence has the property that $\varepsilon_0 \geq \varepsilon_1 \geq \cdots \geq \varepsilon_n = 0$. In other words, the sequence approaches D in the bottleneck distance. Moreover, the triangle inequality then gives a guarantee that for any PD X and all i

$$|\mathrm{d}_B(X, D) - \mathrm{d}_B(X, D_i)| \leq \varepsilon_i.$$

In addition to computing the sequence of diagrams, we will also compute the optimal matching $M_i : D_i \to D_{i+1}$. Thus, given a matching $M : X \to D_i$, we will be able to compute a matching $M_i \circ M : X \to D_{i+1}$ whose bottleneck cost is only increased by at most ε_{i+1}.

We will show how this can be used to efficiently reuse much of the work in computing a bottleneck distance to D_i when computing the distance to the finer approximation D_{i+1}. Surprisingly, the total space required to represent the sequence of diagrams as well as the sequence of matchings will only be $O(n)$. So, for a constant factor extra space, the PD can be stored in a way that allows for fast approximate distance computations.

Our approach is based on a greedy permutation (also known as a farthest-point traversal) of the points in the persistence diagram combined with a weighting scheme. For a PD with n points, we will produce n different approximations, where the ith approximation has i distinct weighted points. During construction, we precompute both the bottleneck distance between the successive approximations, but also the matching that realizes this distance. Thus, we have bounds on the error associated to any given approximation.

The main motivation for our work is to improve data analysis on spaces of persistence diagrams. In many proximity search problems, it is often less important to compute distances than it is to guarantee that distances are larger than some bound in order to prune a search. In simple metric spaces where distances are inexpensive to compute or are part of the input, this is usually accomplished by direct comparison. For expensive to compute metrics such as the bottleneck distance, we would benefit from fast approximations.

2 Background

The extended plane is the set $\mathbb{R}^2_\infty := (\mathbb{R} \cup \{\infty\})^2$. Let $p = (p_b, p_d)$ be a point in $\mathbb{R}^2_\infty$. The subscripts b and d on the coordinates refer to "birth" and "death" respectively. The distance between points $p, q \in \mathbb{R}^2$ is defined using the L_∞-norm

$$\|p - q\| = \max\{|p_b - q_b|, |p_d - q_d|\}.$$

The distance between between points in $\mathbb{R}^2_\infty$ is defined similarly with the usual care required for arithmetic on ∞.

A persistence diagram (PD) is a multiset of points in $\mathbb{R}^2_\infty$ that contains the diagonal $\{(x, x) \mid x \in \mathbb{R}^\infty\}$ with infinite multiplicity. For any multiset X, let $\underline{X}$ denote the underlying set, and for any $x \in \underline{X}$, let $\text{mult}(x)$ be the multiplicity of x in X.

Let A and B be PDs. A bijection $M : A \to B$ is called a *matching*. The *bottleneck cost* of M is $\max_{a \in A} \|a - M(a)\|$. The *bottleneck distance* between PDs A and B is defined to be the minimum bottleneck cost over all matchings $M : A \to B$.

Every matching $M : A \to B$ induces a *transportation plan*. This is a function $T : \underline{A} \times \underline{B} \to \mathbb{Z}$ that counts the number of edges between a point $a \in \underline{A}$ and $b \in \underline{B}$. More generally a function $T : \underline{A} \times \underline{B} \to \mathbb{Z}$ is a transportation plan between multisets A and B if for all $a \in \underline{A}$ and all $b \in \underline{B}$, we have

$$\sum_{b' \in \underline{B}} T(a, b') = \text{mult}(a) \text{ and } \sum_{a' \in \underline{A}} T(a', b) = \text{mult}(b).$$

The bottleneck cost of a matching is easily computed from the transportation plan. For this reason, it is not uncommon to see the bottleneck distance presented in terms of either. Both matchings and transportation plans will be useful in this paper. Matchings have the advantage that their composition is canonically defined. Transportation plans have the advantage that they are simpler to represent and therefore reduce space usage. Every transportation plan represents an equivalence class of matchings.

The bottleneck distance is a special case of the Wasserstein distance. For a given matching M, the p-Wasserstein cost is

$$\text{cost}_p(M) = \left(\sum_{x \in X} \mathrm{d}(x, M(p))^p \right)^{1/p}.$$

The bottleneck distance is the case of $p = \infty$.

2.1 Greedy Permutations

Given two subsets A and B in a metric space, the *Hausdorff distance* between A and B is defined to be

$$\mathrm{d}_H(A, B) := \max\{\sup_{a \in A} \inf_{b \in B} \mathrm{d}(a, b), \sup_{b \in B} \inf_{a \in A} \mathrm{d}(a, b)\}.$$

Let (X, d) be any metric space and let $P \subseteq X$ be finite. Let P be ordered $(p_0, \ldots, p_{n-1})$. The ith *prefix* of the ordering is denoted $P_i = \{p_0, \ldots, p_{i-1}\}$. The ordering is a *greedy permutation* if for all $i \in \{1, \ldots, n-1\}$,

$$\min_{q \in P_i} \mathrm{d}(p_i, q) = \mathrm{d}_H(P_i, P).$$

In other words, each point p_i is the farthest point from the prefix P_i. For this reason, greedy permutations are also known as farthest point samples. The point p_0 may be chosen arbitrarily.

For each p_i, the distance $\varepsilon_i = \mathrm{d}(p_i, P_i)$ is called the *insertion radius*. By convention $\varepsilon_0 = \infty$. By construction, for a greedy permutation, we have

$$\varepsilon_0 \geq \varepsilon_1 \geq \cdots \geq \varepsilon_{n-1}.$$

The *spread* Δ of a point set is the ratio of the largest to smallest pairwise distances. If the points are arranged in a greedy permutation, then the spread is at most $\frac{2\varepsilon_1}{\varepsilon_n}$. We define the spread for a persistence diagram similarly, except that we ignore multiplicity and treat the entire diagonal as a single point.

3 Related Work

The state-of-the-art for computing the bottleneck distance between persistence diagrams is the work of Kerber et al. [17]. They borrow the geometric insight from the work of Efrat et al. [12], in which the Hopcroft-Karp matching algorithm [16] is combined with geometric data structures to avoid constructing the entire bipartite graph. With this approach Efrat et al. reduce the execution time of the matching algorithm from $O(n^{2.5})$ to $O(n^{1.5} \log n)$. Kerber et al. use this idea to give a similar running time for computing the bottleneck distance of persistence diagrams.

Kerber et al. also adapt Bertsekas' auction algorithm [2] to find an approximate Wasserstein distance between diagrams, including the adaptation by Bertsekas and Castanon [3] that works with multiplicity.

Our approach is based on greedy permutations of the points as originally presented for clustering (see [14, 11]). An efficient algorithm to compute such permutations comes from the work of Clarkson [8] and his data structures for nearest neighbor search. Har-Peled

and Mendel [15] showed that Clarkson's algorithm runs in $2^{O(d)}n\log(\Delta)$ time and $O(2^{O(d)}n)$ space, where d is the doubling dimension and Δ is the spread. Our work extends Clarkson's algorithm and uses it to compute greedy permutations of PDs.

There are many novel applications of the bottleneck and Wasserstein distance, or its approximation. Fasy et al. [13] consider the problem of approximating nearest neighbors of PDs. Mumey [19] provides an approach to approximate nearest bottleneck distance queries over a finite point set by indexing a set of points to create a database and a *trie*-based data structure. Soler et al. [22] track topological changes in time by finding matchings in a lifted Wasserstein metric. Vidal et al. [23] provide an iterative algorithm to calculate progressively accurate approximations of the Wasserstein barycenters on a space of PDs. All of these approaches could benefit from faster distance computation.

There are several previous approaches that transforming PDs into another representation to make certain tasks computationally simpler. Bubenik [4] introduces one such form called persistence landscape which enables statistical inference via standard statistical tests. Adams et al. [1] show that PDs can be converted to a vector representation called a persistence image. Divol and Lacombe [10] give a framework to study PDs by converting them to partial optimal transport problems and expressing them as Radon measures on the upper half plane. Lacombe et al. [18] describe a scalable framework to compute standard properties of PDs by reformulating them as optimal transport problems.

Additionally, there has been significant work in representing PDs as vectors leading to a combination of machine learning theory with topological data analysis. Carrière et al. [6] define a stable topological point signature for shapes by extracting vectors from PDs. Many machine learning techniques require the underlying space to be a Hilbert space. It is shown by Reininghaus [20] that it is possible to use topological information encapsulated in PDs in all kernel-based machine learning methods. The kernel thus defined, however, performs poorly when encumbered with a large number of training vectors. Zeppelzauer [24] attempts to overcome these limitations by analysing 3D surfaces using persistence image descriptors of Adams et al. [1]. Carrière et al. [5] implement a provably stable sliced Wasserstein kernel and an algorithm to approximate it.

4 From Hausdorff to Bottleneck Approximations

The main idea of this paper is to use greedy permutations to give approximations to a persistence diagram. The greedy permutation is often used to give approximations that are close in Hausdorff distance. If A and B are PDs, then it is possible that $\mathrm{d}_H(\underline{A},\underline{B}) = 0$ and $\mathrm{d}_B(A,B)$ is large. Therefore, one should not, in general, expect that a good Hausdorff approximation will give a good bottleneck approximation. The same holds for other Wasserstein metrics.

There is one important case in which we *can* relate the Hausdorff distance and the Bottleneck distance – when one diagram is a subset of the other and the multiplicities of its points have been carefully adjusted. This section gives the construction and proves the equivalence of the Hausdorff and Bottleneck distances for that case.

Natural Reweighting

A *reweighting* of a PD A is a new persistence diagram A' formed by assigning new multiplicities to the points. Thus, if A is a reweighting of A, then $\underline{A} = \underline{A'}$.

Given $A \subseteq B$, the *natural reweighting* of A with respect to B is one that assigns a multiplicity to each point a in $\underline{A}$ according to the number of points in B having a as their nearest neighbor. That is, for $a \in \underline{A}$, we have $\mathrm{mult}(a)$ is the number of points of B that are closer to a than to any other point of $\underline{A}$. Ties can be broken arbitrarily, possibly resulting in non-uniqueness.

▶ **Lemma 1.** *Let A and B be PDs with $A \subseteq B$. If A' is a natural reweighting of A, then*

$$\mathrm{d}_B(A', B) = \mathrm{d}_H(\underline{A}, \underline{B}).$$

Proof. All points in $A \cap B$ will be matched to themselves. The points of $B \setminus A$ will be matched to their nearest neighbor. This exactly matches each point of A' with the correct multiplicity. This is a matching whose bottleneck is $\max_{b \in B} \mathrm{d}(b, A) = \mathrm{d}_H(A, B)$. This matching must be optimal, because every edge from $b \in B$ is the globally shortest possible to a point in A. ◀

Optimal Transport

The natural reweighting can be understood in terms of optimal transport. Specifically, for $A \subseteq B$, the natural reweighting A' corresponds to the transportation plan that moves every point b of B to its nearest point $\mathrm{NN}_{\underline{A}}(b)$ in $\underline{A}$. Formally, the transportation plan is

$$T(a, b) := \begin{cases} \mathrm{mult}(b) & \text{if } a = \mathrm{NN}_{\underline{A}}(b) \\ 0 & \text{otherwise.} \end{cases}$$

The transportation plan T is optimal for any Wasserstein metric. Any other transportation plan would suffer increased cost for each point of B that is not moved to its nearest neighbor.

5 Greedy Permutations of PDs

Given a PD, the diagonal and the multiplicity of points makes it impossible to give a greedy permutation directly. With a slight adjustment, we can define a greedy permutation of the nondiagonal points of a PD. Let D be a PD and let the nondiagonal points of $\underline{D}$ be ordered $p_0, \ldots, p_{n-1}$. The ith approximate diagram D_i is the natural reweighting of the ith prefix of the ordering with the diagonal added to make it a PD. So, D_0 is the empty diagram consisting of just the diagonal and D_n is D. The ordering is greedy if for all $i \in \{0, \ldots, n-1\}$,

$$\min_{q \in D_i} \|p_i - q\| = \mathrm{d}_H(\underline{D_i}, \underline{D}).$$

The sequence $(D_0, \ldots, D_n)$ is called a *greedy PD sketch* of D.

By always starting with the diagonal, the choice of p_0 is not arbitrary as is the case of greedy permutations in other metrics. The permutation will always start with p_0 as a point of maximum persistence. Also, it is not relevant to consider multiplicities when finding greedy permutations of PDs. Once all the distinct points have been added, the Hausdorff distance will be zero.

Because D_i is the natural reweighting with respect to D, the result is a sequence of bottleneck approximations to D. Up to a factor of two, these approximations are optimal for their size in the following sense.

▶ **Theorem 2.** *Let D be a persistence diagram and let $(D_0, \ldots, D_n)$ be a greedy PD sketch of D. For all i, let Opt_i be the PD with i distinct points that minimizes $\mathrm{d}_B(D, \mathrm{Opt}_i)$. Then, for all i, the approximation D_i satisfies*

$$\mathrm{d}_B(D, D_i) \leq 2\mathrm{d}_B(D, \mathrm{Opt}_i).$$

Proof. The proof follows the same pattern as the standard proof that greedy permutations yield 2-approximate k-centers in metric spaces [14, 11]. Let $r = \mathrm{d}_B(D, \mathrm{Opt}_i)$. Any point of D paired with the diagonal in the optimal matching with Opt_i can also be matched with

the diagonal in the matching with D_i. So, it will suffice to consider points of D matched to nondiagonal points. By construction, the distance between any pair of nondiagonal points in D_i is at least $\min_{q \in D_i} \|p_i - q\| = \mathrm{d}_H(\underline{D_i}, \underline{D})$. So, if any two points $a, b \in D_i$ are matched to the same nondiagonal point q in Opt_i, then, by the triangle inequality,

$$\mathrm{d}_H(\underline{D_i}, \underline{D}) \leq \|a - b\| \leq \|a - q\| + \|b - q\| \leq 2r.$$

So, we may assume that the bottleneck matching that realizes $\mathrm{d}_B(D, \mathrm{Opt}_i)$ matches each point of D_i with a unique point of Opt_i. Using the triangle inequality, every point of D is within $2r$ of a point of D_i. Therefore, we have shown that $\mathrm{d}_H(\underline{D_i}, \underline{D}) \leq 2r$, and so, by Lemma 1, it follows that $\mathrm{d}_B(D_i, D) \leq 2r$. ◀

The preceding theorem shows the sense in which the PD sketch is approximately optimal for its size. When used as a proxy for a full diagram, there is a clear upper bound on the error; using D_i instead of D introduces at most ε_i error as shown in the following lemma.

▶ **Lemma 3.** *Let X and D be persistence diagrams and let i be a nonnegative integer. Let D_i be the ith PD in a greedy PD sketch of D and let $\varepsilon_i = \mathrm{d}_H(\underline{D_i}, \underline{D})$. Then,*

$$|\mathrm{d}_B(X, D) - \mathrm{d}_B(X, D_i)| \leq \varepsilon_i.$$

Proof. Because D_i is the natural reweighting of a subset of D, Lemma 1 implies that $\mathrm{d}_B(D, D_i) = \mathrm{d}_H(\underline{D}, \underline{D_i}) = \varepsilon_i$. The desired inequality, then follows from the triangle inequality for the bottleneck distance. ◀

5.1 Transportation Plans

In addition to the approximate PDs, we also want to compute a matching or a transportation plan that take us from D_i to D_{i+1}. The difference between these PDs is

- the addition of point p_i,
- some mass moves from points in D_i to p_i, and
- some mass moves from the diagonal to p_i.

By the definition of the natural reweighting, the movement of mass is just a count of how many points of D have p_i as their nearest neighbor. By the triangle inequality in the plane, any such transportation plan must be optimal as the source and target of every unit of mass is known and the straight lines have minimal cost.

The natural reweighting of D_{i+1} is with respect to D rather than D_i. As a result, the bottleneck distance between D_i and D_{i+1} may be larger than the bottleneck distance between D_i and D. Specifically, we have

$$\varepsilon_i = \mathrm{d}_H(\underline{D_i}, \underline{D_{i+1}}) \leq \mathrm{d}_B(D_i, D_{i+1}) \leq \varepsilon_i + \varepsilon_{i+1}.$$

The sketch for a PD with n points contains $n + 1$ PDs and n transportation plans. With a little care, this can be represented in $O(n)$ space as shown in the following theorem.

▶ **Theorem 4.** *Given a persistence diagram D with n points, the greedy PD sketch and the optimal transportation plans can be represented in $O(n)$ space.*

Proof. Without the multiplicities, the sequence of diagrams is represented by the greedy permutation of the points. The multiplicities are encoded in the transportation plans from D_i to D_{i+1}. In such a transportation plan, all the mass that moves is shifted to the newly

added point p_i. Say that a point $q \in D_i$ is a neighbor of p_i if it is one of the points that shifts some of its mass to p_i. That is, there is some point $x \in D$ such that $NN_{D_i}(x) = q$ and $NN_{D_{i+1}}(x) = p_i$. By the greedy ordering,

$$\|x - p_i\| < \|x - q\| \leq \varepsilon_i.$$

Moreover, the greedy permutation guarantees that the neighbors of p_i are all ε_i-separated. So, the squares of side-length ε_i centered at the neighbors of p_i are all disjoint and contained in the square of side length $5\varepsilon_i$ centered at p_i. It follows that there are at most 25 neighbors including the diagonal. Thus, the ith transportation plan can be represented by a list of at most 25 neighbors along with an integer for each counting the amount of mass moved. In total this is $O(n)$ numbers to store. ◀

6 Modifying Clarkson Algorithm for Greedy Permutations of Persistence Diagrams

It is possible to compute the greedy permutation of a persistence diagram using the standard quadratic time algorithm [14, 11]. The naive approach is simply to treat the entire diagonal as the first point in the permutation. To get a faster algorithm, a simple and effective approach due to Clarkson can compute a greedy permutation in $O(n \log \Delta)$ time in low-dimensional metrics. However, the inclusion of the diagonal in a persistence diagram is an obstacle to direct application of Clarkson's algorithm. The reason is that treating the diagonal as a point breaks the triangle inequality, e.g., consider two points that are both close to the diagonal but far from each other (see Fig. 1). If one enforced the triangle inequality, the impact would be that the diagram could appear to have high doubling dimension; all n points could be one unit from the diagonal and thus equidistant (exactly two units) from each other. In this section, we will show how to modify the algorithm so that we can achieve the same $O(n \log \Delta)$ running time for persistence diagrams. The main insight is to augment the PD with its projection to the diagonal and modify the way distances are computed. We will also give an example where the direct application of Clarkson's algorithm devolves to quadratic time.

6.1 Overview of Clarkson's Algorithm

In his work on nearest neighbor search in metric spaces [7, 8, 9], Clarkson developed several data structures based on a kind of discrete Voronoi diagram. Points are inserted into the structure one at a time and the uninserted points are assigned to their nearest neighbors among the inserted points. The set of points whose nearest neighbor is a point p are called the *(discrete) Voronoi cell* of p and are denoted $\mathrm{Vor}(p)$. A neighborhood graph is constructed on the inserted points with edges between points p and q representing the possibility that inserting a point in $\mathrm{Vor}(p)$ would alter $\mathrm{Vor}(q)$ or vice versa. As points are inserted, updates to the structure remain local with respect to this neighborhood graph. Some extra edges are maintained to simplify construction, pruning only those between points whose distance is more than some constant times their radii.

The Voronoi cells provide exactly the information required to compute both the natural reweighting as well as the optimal transportation plans. The improvements in running time come from the sparsity of the neighborhood graph. That sparsity will also translate into the sparsity of the PD sketch.

The simplest version of Clarkson's data structure is called *sb* and was implemented in C. It was originally designed to use a random permutation of the input points, but after experimental analysis, it was shown that performance improved when using a greedy

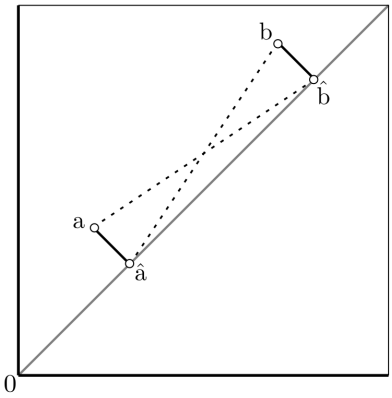


Figure 1 The triangle inequality fails when treating the diagonal as a point. If $\mathrm{d}(a,b) \gg \mathrm{d}(a,\hat{a})$ and $\mathrm{d}(a,b) \gg \mathrm{d}(b,\hat{b})$ then the triangle inequality fails as $\mathrm{d}(a,b) > \mathrm{d}(a,\hat{a}) + \mathrm{d}(\hat{a},b)$.

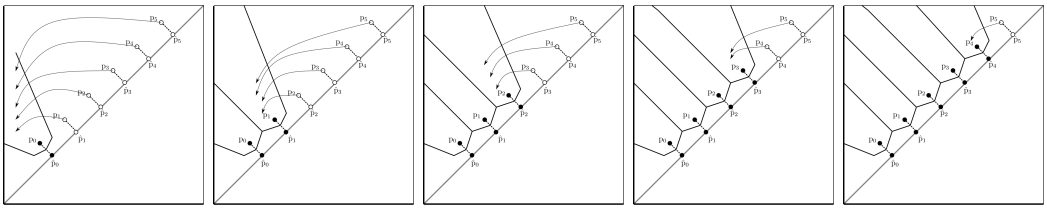


Figure 2 In this example, Clarkson's algorithm devolves to quadratic time. As points are added p_0 to p_5, all other points have to be checked to see if they move.

permutation [8]. The data structure also finds the greedy permutation efficiently as part of the construction: knowing the Voronoi cells of the points added so far makes it easy to add the next farthest point by storing the Voronoi cells in a priority queue. Later, Har-Peled and Medel showed that this algorithm runs in $O(n \log \Delta)$ time in doubling metrics. A Python implementation of this algorithm is also available [21].

Consider a PD in which all the points are on a line one unit from the diagonal and are spaced out two units apart. With a slight perturbation, any given ordering can be the greedy permutation. If the ordering is sorted along the line, then each new point would require checking all the other points to see if they move. Thus the total running time is quadratic, even though the spread is linear. This is illustrated in Fig. 2. The reason this seems to violate the $O(n \log \Delta)$ running time is that the big-O hides a term exponential in the doubling dimension. If the distances are replaced with shortest path distances to enforce the triangle inequality, the doubling dimension of this example becomes $\log n$ because the the path from a point to the diagonal plus the diagonal to another point results in all points equidistant. So, in order to compute the greedy permutation in subquadratic time using Clarkson's algorithm, one must treat the diagonal differently.

6.2 Using Projections

Let D be a persistence diagram. Let X be the nondiagonal points of D and let $\hat{X}$ be the projections of the points of X to the diagonal. For each $a \in X$, we will write $\hat{a}$ for its projection. Ultimately, we will compute the greedy permutation of $X \cup \hat{X}$ and retain only the ordering on X.

Let S be a subset of $X \cup \hat{X}$. The points $\hat{X}$ naturally partition the diagonal into segments where for any $\hat{x} \in S \cap \hat{X}$,

$$\mathrm{seg}_S(\hat{x}) := \{\hat{y} \mid \text{for all } \hat{z} \in S \cap \hat{X}, \|\hat{x} - \hat{y}\| \leq \|\hat{z} - \hat{y}\|\}.$$

When partitioning the points into discrete Voronoi cells, we will consider the Voronoi cells of the segments for the diagonal points. Formally,

$$\mathrm{d}_S(x, y) := \begin{cases} \min_{\hat{z} \in \mathrm{seg}_S(x)} \|\hat{z} - y\| & \text{if } x \in \hat{X} \\ \|x - y\| & \text{otherwise.} \end{cases}$$

This way of computing distances does not give a metric, but it will give the correct notion of discrete Voronoi cells, defined as

$$\mathrm{Vor}_S(x) := \{y \in X \cup \hat{X} \mid \text{ for all } z \in S, \mathrm{d}_S(x, y) \leq \mathrm{d}_S(z, y)\}.$$

Note that the addition of more points of $\hat{X}$ into S will not affect the Voronoi cells of points in $S \cap X$. This is the desired invariant because it means that the union of the Voronoi cells of the diagonal points only depends on the nondiagonal points. The points of $S \cap \hat{X}$ serve only to partition what would otherwise be one large cell associated to the diagonal.

The *projection distance* is

$$r(x, y) := \begin{cases} \|y - \hat{y}\| & \text{if } x \in \hat{X} \text{ and } y \in X \\ \|x - y\| & \text{otherwise.} \end{cases}$$

The *radius* of a Voronoi cell of a point $x \in S$ is

$$\mathrm{rad}(x) := \max_{y \in \mathrm{Vor}_S(x)} r(x, y).$$

The modified Clarkson algorithm then computes the greedy PD sketch by maintaining these Voronoi cells. The next point added at each step is the farthest point (in projection distance) in the Voronoi cell of maximum radius. The neighborhood graph connects two Voronoi cells as long as their distance is less than four times the larger of their radii.

The key to the analysis of Clarkson's algorithm is that the neighborhood graph maintains constant degree. One must keep enough edges so that the following two conditions hold.

1. The Voronoi cell of a newly inserted point is contained in the union of the cells of the neighbors of its parent, i.e., the nearest neighbor just prior to insertion.
2. The neighbors of a newly inserted point are contained in the union of the neighbors of neighbors of its parent just prior to insertion.

Our changes to how distances are computed produce only a small change to the analysis of Clarkson's algorithm, requiring us to store edges in the neighborhood graph up to four times the radius (rather three times the radius as in the original). The full version gives the details on why this small change is sufficient (by repeated use of the triangle inequality).

▶ **Theorem 5.** *The modified Clarkson algorithm restricted to the nondiagonal points gives a greedy PD sketch.*

Proof. It suffices to prove that each time a nondiagonal point p_i is added, it is the farthest point from D_i. The algorithm explicitly chooses the farthest point at each step, so we need only observe that the inclusion of the projections do not change the Voronoi cells of the nondiagonal points. In other words, the projections do not change the order in which the nondiagonal points are added. This follows from the definition of the projection distance. ◀

▶ **Theorem 6.** *The greedy PD sketch can be computed in $O(n \log \Delta)$ time.*

Proof. Using the projections, the analysis of the running time is the same as the standard analysis as given in Har-Peled and Mendel [15] and also in Clarkson [8]. The key idea is to count the number of times any point is considered for moving into a new Voronoi cell. By a volume packing argument, this can happen only a constant number of times before the maxmimum radius goes down by at least a factor of 2. The one caveat is that the projection could artificially increase the spread. This is resolved by stopping the algorithm as soon as all of the nondiagonal points are inserted. This happens at scale $\frac{1}{\Delta}$ times the diameter and therefore, the total running time is $O(n \log \Delta)$ as desired. ◀

A natural improvement in many instances would be to stop early and return a prefix of the PD sketch. In particular, if one has other sources of error – such as the grid size in persistence images [1] – one need not compute the full sketch. For constant precision, the running time will be linear.

▶ **Corollary 7.** *Let D be a PD, and let $R_{\max}$ be the maximum persistence of any point in D. For a fixed precision ε, a partial greedy PD sketch $(D_0, \ldots, D_k)$ can be computed such that*

$$\mathrm{d}_B(D_k, D) \leq \varepsilon$$

in $O(n \log \frac{R_{\max}}{\varepsilon})$.

7 Updating a Matching

Starting from a greedy PD sketch of D and a second PD X, an approximation D_i in the sketch can be used to estimate $\mathrm{d}_B(X, D)$. In doing so, an optimal bottleneck matching between D_i and X is computed. Choosing a larger value of i will increase the precision. In this section, we show how to bootstrap the computation that has already been done for D_i to get good matching between X and D_{i+1}. The principle idea is to compose a matching $X \to D_i$ with a matching $D_i \to D_{i+1}$ that is consistent with the transportation plan T_i. The result will not necessarily be the optimal matching $X \to D_{i+1}$, but it will satisfy the cost guarantee of Lemma 3. Thus, in many cases, most of the work of finding the bottleneck matching will already be done.

Naive Update

The transportation plan $T_i : \underline{D_i} \times \underline{D_{i+1}} \to \mathbb{Z}$ encodes an equivalence class of matchings. Any matching $M_i : D_i \to D_{i+1}$ that has $T_i(q, q')$ edges from q to q' is in this class. One can easily choose such a matching arbitrarily and compose it with the previously computed optimal matching $M : X \to D_i$. That is, $M_i \circ M : X \to D_{i+1}$ is a matching. Because M_i has bottleneck cost at most ε_i, the new matching will increase in cost by at most ε_i (by the triangle inequality).

A Short Auction

It is possible to choose a locally optimal matching update consistent with T_i. This is perhaps most easily understood in terms of the p-Wasserstein cost of the matching. Minimizing this cost is equivalent to minimizing the pth power of the cost. So, replacing one edge xq with an edge xp_i in a matching M' results in the following change.

$$\text{cost}_p(M')^p - \text{cost}_p(M)^p = \text{d}(x, p_i)^p - \text{d}(x, q)^p.$$

Thus, for each such edge xq, we compute the corresponding pth power change in cost. Then, we update the edges in order of this change. This resembles an abbreviated version of the auction algorithm [2].

For bottleneck matchings, one cannot assign costs using the pth power of the distances, because $p = \infty$ in that case. Instead, one can find the optimal matching from the neighbors of q in the matching M to the points q and p_i (with multiplicities). This only requires sorting the $O(n)$ edges by their length. The bottleneck matching can be found in the minimum prefix of this ordering that has sufficiently many edges incident to the points of X and the points q and p_i (by Hall's Theorem). We call the resulting matching, *the locally optimal matching consistent with* T_i.

In either case, the time is bounded by the cost of sorting the points matched to q for each q that shifts some mass to p_i in the transportation plan T_i. We have already shown that there are only at most 25 such points q, but with multiplicity, there could be as many as $|X| = \Theta(n)$ edges to consider. This makes the overall, worst-case running time $O(n \log n)$ for an update to the matching.

Amortizing the Cost of Updates

Although the worst case cost of updating the matching when going from D_i to D_{i+1} is $O(n \log n)$, not every such update can be that expensive. Below, we show that a sequence of these updates from D_i to D_j such that $\varepsilon_i \leq 2\varepsilon_j$ will also only take $O(n \log n)$ time. This means adding points in the sketch to halve the error (double the precision) is asymptotically the same (in the worst-case) as updating the matching for one new point.

▶ **Theorem 8.** *Let $(D_0, \ldots, D_n)$ be a greedy PD sketch of D. For any i, k such that $\varepsilon_i \leq 2\varepsilon_k$, the total cost of computing the locally optimal matchings M_j consistent with T_j for all $i \leq j \leq k$ can be computed in $O(n \log n)$ time.*

Proof. The total cost is $\sum_{q \in Q} \text{mult}(q) \log(\text{mult}(q))$ where Q is the set of points whose mass changes at some point in the sequence of updates. Recall that by the definition of the natural reweighting used in the greedy PD sketch, the multiplicity of a point $q \in D$ is equal to the number of points in a discrete Voronoi cell at some point in the construction. If $T_j(q, p_j) > 0$, then q and p_j have neighboring Voronoi cells in some step of the construction. This means that every point in $\text{Vor}(q)$ is touched when p_j is inserted to see if it must be move. So, as we observed in the analysis of the construction (Theorem 6), any given point can be touched at most $O(1)$ times before the insertion radius decreases by a factor of 2. As the edges incident to any point q in the matching are in correspondence with the points of $\text{Vor}(q)$, it follows that each such edge participates in at most $O(1)$ updates. In other words, $\sum_{q \in Q} \text{Vor}(q) = \sum_{q \in Q} \text{mult}(q) = O(n)$. So, the total cost is

$$\sum_{q \in Q} \text{mult}(q) \log(\text{mult}(q)) \leq \sum_{q \in Q} \text{mult}(q) \log(n) = O(n \log n). \qquad \blacktriangleleft$$

8 Filtered Neighborhood Graphs

Matchings are computed on neighborhood graphs. In this section, we show how the greedy sketch computation can also simplify the construction of these graphs. We then show how this same construction allows for fast Hausdorff approximation between PD sketches, leading to fast lower bounds on the bottleneck distance.

The *α-neighborhood graph* on a set P is the graph

$$\mathrm{Nbrhd}(P, \alpha) := (P, \{(p_i, p_j) \mid \mathrm{d}(p_i, p_j) \leq \alpha\}).$$

Let λ and γ be constants. If the points of P are all pairwise λ apart, then the degree of any vertex in $\mathrm{Nbrhd}(P, \gamma\lambda)$ cannot exceed $(2\gamma + 1)^2$. This follows because the squares of side length λ centered at the neighbors will be disjoint and contained in the square of side length $(2\gamma + 1)\lambda$.

Let $P = (p_0, \ldots, p_{n-1})$ be a set ordered according to a greedy permutation. The *γ-filtered neighborhood graph* on P is the graph with vertices P and edges (p_j, p_i) whenever $i < j$ and $\mathrm{d}(p_i, p_j) < \gamma\lambda_j$. Because P_j is a λ_j-net for all j, there will be at most $(2\gamma + 1)^2$ neighbors of p_j that precede it in the greedy permutation. Thus, the total number of edges is $(2\gamma + 1)^2 n$. Moreover, the graph contains $\mathrm{Nbrhd}(P_i, \gamma\lambda_i)$ for all i.

When computing the greedy permutation, one can compute the filtered neighborhood graph at the same time in the same asymptotic running time. This is the underlying idea in the Clarkson algorithm (the graph is an undirected version of the *sb* data structure).

For a pair of compact sets P, Q, the bipartite α-neighborhood graph is

$$\mathrm{BiNbrhd}(P, Q, \alpha) := (P \sqcup Q \mid \{(p, q) \in P \times Q \mid \mathrm{d}(p, q) \leq \alpha)$$

The Hausdorff distance between P and Q is the minimum α such that $\mathrm{BiNbrhd}(P, Q, \alpha)$ contains no isolated vertices. The bottleneck distance between P and Q is the minimum α such that $\mathrm{BiNbrhd}(P, Q, \alpha)$ contains a perfect matching. If P and Q are persistence diagrams, one adds the diagonal as an extra vertex to each set and adds edges from points to the diagonal if their projection to the diagonal is within α.

The main way that "geometry helps" for matching problems in the plane is that one can use a geometric data structure to implicitly store this graph. However, when working with approximations, one can show that the bipartite neighborhood graph has linear size and can be computed in linear time if one has already precomputed the filtered neighborhood graphs of the two sets. Below, we explain the construction.

▶ **Lemma 9.** *Let $R, B \subset \mathbb{R}^2$. If $\mathrm{BiNbrhd}(R_\lambda, B_\lambda, \gamma\lambda)$ has an isolated vertex, then $\mathrm{d}_H(R, B) \geq \lambda(\gamma - 1)$.*

Proof. Without loss of generality, let $x \in R_\lambda$ be an isolated vertex. Then, there are no points of B_λ within distance $\lambda\gamma$ of x and so, $\mathrm{d}_H(R, B_\lambda) \geq \lambda\gamma$. Because B_λ is a λ-net, $\mathrm{d}_H(B, B_\lambda) \leq \lambda$. Therefore, by the triangle inequality, $\mathrm{d}_H(R, B) \geq \lambda\gamma - \lambda = \lambda(\gamma - 1)$. ◀

▶ **Theorem 10.** *Let R, B be PDs and let λ and γ be constants such that $\lambda(\gamma - 1) \geq \mathrm{d}_H(R, B)$. Given the greedy permutations of R and B as well as the corresponding $(2\gamma + 1)$-filtered neighborhood graphs, the $\mathrm{BiNbrhd}(R_\lambda, B_\lambda, \gamma\lambda)$ can be computed in linear time.*

Proof. The algorithm will be incremental, adding the points in order of their insertion radii. At each step i, we maintain $G_i = \mathrm{BiNbrhd}(R_{\lambda_i}, B_{\lambda_i}, \gamma\lambda_i)$, where λ_i is the insertion radius of the newly inserted point. When inserting p_i, we add its neighbors G_i. Without loss of generality, assume $p_i \in R$. Let y be the nearest neighbor of p_i in $\mathrm{Nbrhd}(R_i, 2\gamma(\lambda_i))$. So

$\mathrm{d}(y, p_i) = \lambda_i$. There are only a constant number of neighbors. Then, let a be any neighbor of y in G_i (which are contained in the neighbors of y in G_{i-1}). The neighbor a must exist because of Lemma 9 and our choice of λ. By the triangle inequality,

$$\mathrm{d}(a, b) \leq \mathrm{d}(a, y) + \mathrm{d}(y, x) + \mathrm{d}(x, b) \leq \lambda_i\gamma + \lambda_i + \lambda_i\gamma = \lambda_i(2\gamma + 1).$$

It follows that a and b are neighbors in $\mathrm{Nbrhd}(B_{\lambda_i}, \lambda_i(2\gamma + 1))$. So, the neighbors of p_i can all be found among the neighbors of y. Because the degrees are constant and edges that are too long for the current value of λ_i can be deleted as they are encountered the neighbors of p_i can be found in amortized constant time.

One pass over the edges suffices to remove any that are longer than $\lambda\gamma$. Thus, the total running time is linear. ◀

▶ **Theorem 11.** *Given the greedy permutations of R and B as well as the corresponding $(2\gamma + 1)$-filtered neighborhood graphs, an approximation of $\mathrm{d}_H(R, B)$ to within a factor of $1 \pm \frac{1}{\gamma}$ can be computed in linear time.*

Proof. If the algorithm from Theorem 10 is run until it either inserts all the points or discovers an isolated vertex, it will produce a graph with a linear number of edges. By iterating over the edges, one can find, for each vertex, the distance to the nearest adjacent vertex in the graph. The maximum of these distances indicates the distance $r = \lambda\gamma$ at which a vertex becomes isolated and is the desired approximation. By Lemma 9, we know that $\mathrm{d}_H(R, B) \geq \lambda(\gamma - 1) = r(1 - \frac{1}{\gamma})$. Similarly, because $\mathrm{d}_H(R, B_\lambda) \leq \lambda$, the triangle inequality implies that $\mathrm{d}_H(R, B) \leq \mathrm{d}_H(R, B\lambda) + \lambda = \lambda(\gamma + 1) = r(1 + \frac{1}{\gamma})$. ◀

9 Conclusions and Future Work

We have presented an efficient method to preprocess a persistence diagram so that one can extract guaranteed approximations with any number of distinct points. It remains to explore the relationship between greedy PD sketches and other PD simplicifcations such as persistence landscapes [4] or persistence images [1]. In particular, it is possible to construct a persistence landscape or a persistence image from a PD sketch, possibly much faster than with the entire diagram. Another application where many PD distance computations are used is in the computation of Wasserstein barycenters. In Vidal et al. [23], a kind of sketch is used to dramatically speed up the Wasserstein barycenter computation. In that case, they use the subset of points of highest persistence. Although the approximation guarantees we prove are only applicable to the bottleneck distance, it seems reasonable that they should also be applicable to other Wasserstein metrics. Indeed, we give the matching update for these metrics in Section 7.

In future work, we will incorporate these sketches into a data structure that supports standard proximity search queries including (approximate) nearest neighbor search, metric range search, and metric range counting. This is the main target of our work as it is a case where there is immediate benefit to finding fast approximate distances with bounds on the error to prune a search.

References

1 Henry Adams, Tegan Emerson, Michael Kirby, Rachel Neville, Chris Peterson, Patrick Shipman, Sofya Chepushtanova, Eric Hanson, Francis Motta, and Lori Ziegelmeier. Persistence images: A stable vector representation of persistent homology. *J. Mach. Learn. Res.*, 18(1):218–252, 2017.

2 Dimitri P. Bertsekas. The auction algorithm: A distributed relaxation method for the assignment problem. *Annals of Operations Research*, 14(1):105–123, 1988.

3 Dimitri P. Bertsekas and David A. Castanon. The auction algorithm for the transportation problem. *Annals of Operations Research*, 20(1):67–96, December 1989. doi:10.1007/BF02216923.

4 Peter Bubenik. Statistical topological data analysis using persistence landscapes. *The Journal of Machine Learning Research*, 16(1):77–102, 2015.

5 Mathieu Carrière, Marco Cuturi, and Steve Oudot. Sliced Wasserstein kernel for persistence diagrams. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 664–673, International Convention Centre, Sydney, Australia, 06–11 August 2017. PMLR. URL: http://proceedings.mlr.press/v70/carriere17a.html.

6 Mathieu Carrière, Steve Y. Oudot, and Maks Ovsjanikov. Stable topological signatures for points on 3d shapes. *Computer Graphics Forum*, 34(5):1–12, August 2015. doi:10.1111/cgf.12692.

7 Kenneth L. Clarkson. Nearest neighbor queries in metric spaces. *Discrete & Computational Geometry*, 22(1):63–93, 1999.

8 Kenneth L. Clarkson. Nearest neighbor searching in metric spaces: Experimental results for "sb(s)". Preliminary version presented at ALENEX99, 2003.

9 Kenneth L. Clarkson. Nearest-neighbor searching and metric space dimensions. In Gregory Shakhnarovich, Trevor Darrell, and Piotr Indyk, editors, *Nearest-Neighbor Methods for Learning and Vision: Theory and Practice*, pages 15–59. MIT Press, 2006.

10 Vincent Divol and Theo Lacombe. Understanding the topology and the geometry of the space of persistence diagrams via optimal partial transport. *Journal of Applied and Computational Topology*, 2020.

11 M.E. Dyer and A.M. Frieze. A simple heuristic for the p-centre problem. *Operations Research Letters*, 3(6):285–288, 1985.

12 A. Efrat, A. Itai, and M. J. Katz. Geometry helps in bottleneck matching and related problems. *Algorithmica*, 31(1):1–28, September 2001. doi:10.1007/s00453-001-0016-8.

13 Brittany Terese Fasy, Xiaozhou He, Zhihui Liu, Samuel Micka, David L. Millman, and Binhai Zhu. Approximate nearest neighbors in the space of persistence diagrams. *CoRR*, abs/1812.11257, 2018. arXiv:1812.11257.

14 Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theor. Comput. Sci.*, 38:293–306, 1985.

15 Sariel Har-Peled and Manor Mendel. Fast construction of nets in low-dimensional metrics and their applications. *SIAM Journal on Computing*, 35(5):1148–1184, January 2006. doi:10.1137/S0097539704446281.

16 John E. Hopcroft and Richard M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2(4):225–231, December 1973. doi:10.1137/0202019.

17 Michael Kerber, Dmitriy Morozov, and Arnur Nigmetov. Geometry helps to compare persistence diagrams. *ACM Journal of Experimental Algorithmics*, 22(1):1.4:1–1.4:20, 2017.

18 Theo Lacombe, Marco Cuturi, and Steve OUDOT. Large scale computation of means and clusters for persistence diagrams using optimal transport. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31, pages 9770–9780. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper/2018/file/b58f7d184743106a8a66028b7a28937c-Paper.pdf.

19 Brendan Mumey. Indexing point sets for approximate bottleneck distance queries. *CoRR*, abs/1810.09482, 2018. arXiv:1810.09482.

20 Jan Reininghaus, Stefan Huber, Ulrich Bauer, and Roland Kwitt. A stable multi-scale kernel for topological machine learning. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, page 4741–4748. IEEE, June 2015. doi:10.1109/CVPR.2015.7299106.

21 Donald R. Sheehy. greedypermutations, 2020. URL: https://github.com/donsheehy/greedypermutation.

22 Maxime Soler, Melanie Plainchault, Bruno Conche, and Julien Tierny. Lifted wasserstein matcher for fast and robust topology tracking. In *2018 IEEE 8th Symposium on Large Data Analysis and Visualization (LDAV)*, page 23–33. IEEE, October 2018. doi:10.1109/LDAV.2018.8739196.

23 Jules Vidal, Joseph Budin, and Julien Tierny. Progressive wasserstein barycenters of persistence diagrams. *IEEE Transactions on Visualization and Computer Graphics*, 26:151–161, 2020.

24 Matthias Zeppelzauer, Bartosz Zieliński, Mateusz Juda, and Markus Seidl. *Topological Descriptors for 3D Surface Analysis*, volume 9667 of *Lecture Notes in Computer Science*, page 77–87. Springer International Publishing, 2016. doi:10.1007/978-3-319-39441-1_8.

A Sparse Delaunay Filtration

Donald R. Sheehy
North Carolina State University, Raleigh, NC, USA

Abstract

We show how a filtration of Delaunay complexes can be used to approximate the persistence diagram of the distance to a point set in $\mathbb{R}^d$. Whereas the full Delaunay complex can be used to compute this persistence diagram exactly, it may have size $O(n^{\lceil d/2 \rceil})$. In contrast, our construction uses only $O(n)$ simplices. The central idea is to connect Delaunay complexes on progressively denser subsamples by considering the flips in an incremental construction as simplices in $d+1$ dimensions. This approach leads to a very simple and straightforward proof of correctness in geometric terms, because the final filtration is dual to a $(d+1)$-dimensional Voronoi construction similar to the standard Delaunay filtration. We also, show how this complex can be efficiently constructed.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Delaunay Triangulation, Persistent Homology, Sparse Filtrations

Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.58

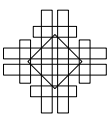
Related Version *Full Version*: https://arxiv.org/abs/2012.01947

Funding This research was supported by the NSF under grant CCF-2017980.

1 Introduction

The persistent homology of the distance to a set of points P in $\mathbb{R}^d$ describes the evolution of the topology of $\bigcup_{p\in P} \text{ball}(p,\alpha)$ as α grows from 0 to ∞. It is a multi-scale description of the "shape" of the point set. The theory of persistent homology has its origins in the work by Edelsbrunner et al. [17, 15] on α-hulls and α-shapes and their relation to the Delaunay triangulation. The paper that introduced persistent homology [18] was based on ordering the simplices of the Delaunay triangulation. Many papers have followed that use alternatives to the Delaunay triangulation in different spaces, but in Euclidean space, the Delaunay triangulation has a certain perfection in its ability to represent the topology of the distance function. For approximations, the potential $O(n^{\lceil d/2 \rceil})$ size of the Delaunay triangulation cannot compete with the linear size of so-called sparse filtrations [35, 2, 6, 13]. In this paper, we combine the ideas from sparse filtrations with the Delaunay triangulation, achieving both the elegance of the Delaunay triangulation and the worst-case linear-size guarantees. Along the way, we will give a new topological perspective to the classic approach to computing Delaunay triangulations by flips.

A *flip* in a 2-dimensional triangulation is the replacement of two adjacent triangles whose four vertices are in convex position with the other two possible triangles on the same vertices. A classic way of visualizing flips is to view the two configurations as projections of the upper and lower hull of a tetrahedron in three dimensions (see Fig. 1). This view also permits one to interpret other operations as flips, such as the insertion of a new vertex splitting one triangle into three. We call the former class of flips $(2,2)$-flips and the latter $(1,3)$-flips, indicating the number of triangles before and after the flip. More generally, there are $(k, d+2-k)$-flips for sets of $d+2$ points in $\mathbb{R}^d$. These are likewise interpreted as projections of $(d+1)$-simplices. In this paper, we will use the $(d+1)$-simplices of the flips to give a topological connection between the Delaunay triangulation of a set of points and the Delaunay triangulation of a subset – both triangulations are subcomplexes of a $(d+1)$-dimensional complex containing the flip simplices. Throughout, we will distinguish between the terms Delaunay triangulation



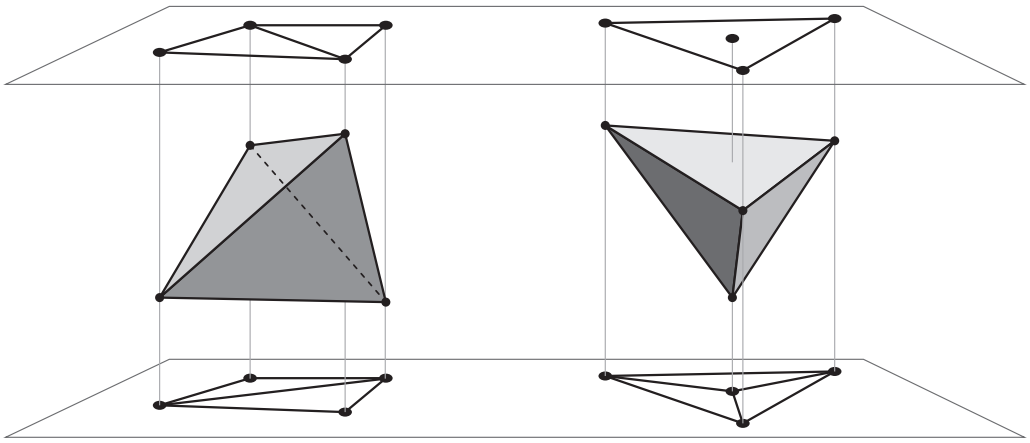


Figure 1 Flips in the plane correspond to the upper and lower facets of a tetrahedron in $\mathbb{R}^3$.

for the embedded geometric complex on a set of points and a Delaunay complex, which is the corresponding abstract simplicial complex. As will become clear, the addition of the flip simplices will result in a simplicial complex that will not be embedded in $\mathbb{R}^d$.

Many Delaunay triangulation algorithms use flips as an algorithmic primitive. This idea goes back to the work of Lawson [28, 27] and reached its more modern form in the simultaneous papers of Bowyer [3] and Watson [37]. As the d-simplices of the Delaunay triangulation are those for which the circumsphere is empty of other input points, a transformations of the problem into $\mathbb{R}^{d+1}$ can make this criterion linear. This was first done by Brown [4] using the stereographic map and later by Edelsbrunner and Seidel [19] using the parabolic lifting. Adjusting the parabolic lifting can be interpreted as assigning weights to points and leads to *weighted Delaunay triangulations* (also known as *regular* triangulations). Edelsbrunner and Shah showed that incremental, flip-based algorithms also work in the weighted case [20] despite obstructions discovered by Joe [26] to applying Lawson's algorithm in $\mathbb{R}^3$.

The theory of persistent homology applies to settings much more general than the sublevel sets of distance functions in Euclidean space. One common setting is to consider points in finite metric spaces. The complex used for this is called the (Vietoris-)Rips complex. At scale α, it contains a simplex for every clique in the α-neighborhood graph of the points. In order to deal with the size blowup of this complex, several sparsification methods have been proposed [35, 2, 6, 13]. All of these approaches attempt to use only subsets of the input points as the scale increases. In this paper, we will show how to adapt this approach to the Delaunay complex.

In other related work, several authors have given sparse approximations for distance functions in Euclidean space. For example, Hudson et al. [25] use ideas from mesh generation to give an approximation to the distance function. Choudhary et al. [11] used a method of overlapping grids to give an approximation that has an asymptotically smaller dependence on the dimension than previous methods.

2 Background

2.1 Distances, Points, and Weights

Let $\|a-b\|$ denote the Euclidean distance between a and b in $\mathbb{R}^d$. Let $\text{ball}(c,r)$ denote the closed ball centered at $c \in \mathbb{R}^d$ with radius r. For a set $P \subset \mathbb{R}^d$, let

$$\mathbf{d}(x,P) = \min_{p\in P} \|x-p\|.$$

Equivalently, $\mathbf{d}(x,P)$ is the minimum r such that $P \cap \text{ball}(x,r)$ is nonempty.

The distance function induced by a point set P maps each point $x \in \mathbb{R}^d$ to $\mathbf{d}(x,P)$. The sublevel sets of this distance function is often used in topological data analysis as a way to extend a finite set and fill in the space between the points. These sublevel sets are sometimes called *offsets* and are formally defined for each scale α as

$$P^\alpha := \{x \in \mathbb{R}^d \mid \mathbf{d}(x,P) \le \alpha\} = \bigcup_{p\in P} \text{ball}(p,\alpha).$$

The Hausdorff distance between two point sets P and Q is defined as

$$\mathbf{d}_H(P,Q) := \max\{\max_{p\in P} \mathbf{d}(p,Q), \max_{q\in Q} \mathbf{d}(q,P)\}.$$

Equivalently, $\mathbf{d}_H(P,Q)$ is the minimum r such that $P \subseteq Q^r$ and $Q \subseteq P^r$.

One way to modify a distance function and give more or less importance to certain points is to assign a weight to each point. A weighted point $\hat{p}$ is a point $p \in \mathbb{R}^d$ and a weight $w_p \in \mathbb{R}$. The *weighted distance to* $\hat{p}$ is defined as

$$\pi_p(x) := \sqrt{\|x-p\|^2 + w_p^2}.$$

The weighted distance is also called the power distance, especially in the case where one subtracts the weight rather than adds it. Throughout the paper, we add weights rather than the usual power distance as it substantially simplifies both the conceptual use of weights to decrease the importance (i.e., radius) of some points and also the arithmetic. All of the constructions we present can be translated into power distances by a global transformation of all the weights and a reinterpretation of the scale. A similar approach to weighting point may be found in [5].

We use the same notation for the weighted distance of a (non-weighted) point x to a weighted point set $\hat{P}$ as we did in the unweighted case.

$$\mathbf{d}(x,\hat{P}) := \min_{\hat{p}\in\hat{P}} \pi_{\hat{p}}(x).$$

Thus, unweighted points way be viewed as points with weight zero. The offsets of $\hat{P}$ are

$$\hat{P}^\alpha = \{x \in \mathbb{R}^d \mid \mathbf{d}(x,\hat{P}) \le \alpha\}.$$

2.2 Persistent Homology

A family of sets $\{X^\alpha \mid \alpha \in \mathbb{R}\}$ is called a *filtration* if for all $\alpha \le \beta$, we have $X^\alpha \subseteq X^\beta$. We will reserve superscripts on sets as a notation for filtration parameters and will denote a filtration (X^α) with parentheses to stress the importance of the ordering. For filtrations that are defined only over an interval $[s,t] \subset \mathbb{R}$, we will assume that $X^\alpha = X^s$ for $\alpha < s$ and $X^\alpha = X^t$ for $\alpha > t$.

The *persistent homology* of (X^α) is a representation of the changes in the topology of X^α as α varies over $\mathbb{R}$. The result is a *persisence diagram*, denoted $\mathrm{Dgm}(X^\alpha)$, that is a multiset of pairs (b, d) in the extended plane $(\mathbb{R} \cup \infty)^2$. Each pair (b, d) represents a nontrivial homology class that exists only in X^α for α in the half open interval $[b, d)$. Thus, b is the *birth time* of a topological feature and d is its *death time*.

Persistent homology is usually computed on combinatorial objects called simplicial complexes. A *simplicial complex* is a pair of sets (V, S) where V is the vertex set and $S \subseteq \mathrm{Pow}(V)$ is the simplex set. It is required that S is closed under subsets, i.e., if $\sigma \subseteq \tau \in S$, then $\sigma \in S$. A family (K^α) of simplicial complexes is called a *filtered simplicial complex* if for all $\alpha \leq \beta$, we have K^α is a subcomplex of K^β.

The main problem addressed in this paper is the efficient approximation of $\mathrm{Dgm}(P^\alpha)$ by constructing a linear-size filtered simplicial complex based on the Delaunay triangulation.

2.3 Greedy Permutations

For ranges of indices, let $[a : b]$ denote $\{a, \ldots b - 1\}$ and $[b]$ denote $[0 : b]$. If the input set P is ordered as $(p_0, \ldots, p_{n-1})$, let $P_i\{p_j \mid 0 \leq j < i\}$ denote the ith prefix of the ordering. The *spread* Δ of P is the ratio of the largest and smallest pairwise distances among points of P.

A *greedy ordering* or *greedy permutation* of P is defined as follows. The first point, p_0, may be chosen arbitrarily. The ith point, p_i is chosen to be a point that maximizes $\mathbf{d}(p_i, P_i)$. That is, each point after the first is the farthest from its predecessors. Equivalently,

$$\mathbf{d}(p_i, P_i) = \mathbf{d}_H(P_i, P).$$

Greedy permutations have been reinvented several times, especially in the context of k-center clustering (see Gonzalez [21] or Dyer and Frieze [14]). Clarkson [12] adapted his sb data structure for nearest neighbor search to compute greedy permutations. Har-peled and Mendel [23] showed that Clarkson's approach yields an $O(n \log \Delta)$-time algorithm in low-dimensional metric spaces. They also gave an $O(n \log n)$-time algorithm in such cases.

The *insertion radius* of a point p_i is defined as

$$r_i := \mathbf{d}(p_i, P_i)$$

By convention, $r_0 = \infty$. For greedy orderings, if $i < j$, then $r_i \geq r_j$.

Every prefix P_i of a greedy permutation satisfies both packing and covering conditions in the following sense. The set P_i is a r_{i-1}-*packing*: for every pair of points in $a, b \in P_i$, we have $\|a - b\| \geq r_{i-1}$. The set P_i is an r_i-*covering* of P: for every point $a \in P$, there is a point $b \in P_i$ such that $\|a - b\| \leq r_i$.

2.4 Weights and Time

We will be considering weighted point sets in which the weights vary in time. For each point p_i in P, we will assign a nonnegative weight function $w_i : \mathbb{R} \to \mathbb{R}$. For a given α, the set

$$\hat{P}(\alpha) := \{(p_i, w_i(\alpha)) \mid p_i \in P\}$$

is a weighted point set in which the weight of p_i is $w_i(\alpha)$. Let $\pi_{i,\alpha}(x)$ denote the weighted distance from the weighted point $(p_i, w_i(\alpha))$ to x.

The weight functions will be defined in terms of the *freezing time* λ_i of each point p_i. The exact value chosen for λ_i will depend on our desired approximation guarantees and the specifics of the algorithm. Once the freezing times are fixed, the weights are as follows.

$$w_i(\alpha) = \begin{cases} 0 & \text{if } \alpha < \lambda_i \\ \sqrt{\alpha^2 - \lambda_i^2} & \text{otherwise} \end{cases}$$

The ball $b_i(\alpha)$ centered at p_i with weight $w_i(\alpha)$ is

$$b_i(\alpha) = \{x \in \mathbb{R}^d \mid \pi_{i,\alpha}(x) \leq \alpha\} = \text{ball}(p_i, \min\{\alpha, \lambda_i\}).$$

This is why λ_i is called the freezing time; at scales $\alpha > \lambda_i$, the Euclidean radius of a ball of weighted radius α will not grow.

The following lemma shows how weighting the points according to the freezing time guarantees that at all scales, there is always a point nearby that is sufficiently close and sufficiently far from its freezing time. The proof can be found in the full version.

▶ **Lemma 1.** *Let $P \subset \mathbb{R}^d$ be ordered according to a greedy permutation with insertion radii $r_0 \ldots r_{i-1}$. Let $\varepsilon > 0$ be any constant. For all $j \in [1:n]$, let the freezing times λ_j be chosen so that $\lambda_j \geq \frac{1+\varepsilon}{\varepsilon} r_j$. Then, for all $k \in [1:n]$ and all $\alpha \geq 0$, there exists i such that*

- $\lambda_i \geq (1+\varepsilon)\alpha$, *and*
- $\|p_i - p_k\| < \varepsilon\alpha$.

In the special case of $\alpha = \lambda_k$ in the preceding lemma, $\text{ball}(p_k, \lambda_k)$ is completely covered by $\text{ball}(p_i, (1+\varepsilon)\lambda_k)$, where the point p_i is not yet frozen at time $(1+\varepsilon)\lambda_k$. For larger values of α, there will always be a point p_i to cover this ball. This lemma has two important consequences. First, it implies that we will be able to remove or ignore the point p_k after time $(1+\varepsilon)\lambda_k$. Second, it allows us to relate the offsets of P with the weighted offsets as shown in the following lemma whose proof may be found in the full paper.

▶ **Lemma 2** (Weighted Offset Interleaving). *Let $P \subset \mathbb{R}^d$ be ordered according to a greedy permutation with insertion radii $r_0 \ldots r_{i-1}$. Let $\varepsilon > 0$ be any constant. For all $j \in [1:n]$, let the freezing times λ_j be chosen so that $\lambda_j \geq \frac{1+\varepsilon}{\varepsilon} r_j$. Then, for all $\alpha \geq 0$,$\hat{P}^\alpha \subseteq P^\alpha \subseteq \hat{P}^{(1+\varepsilon)\alpha}$.*

2.5 Delaunay and Voronoi

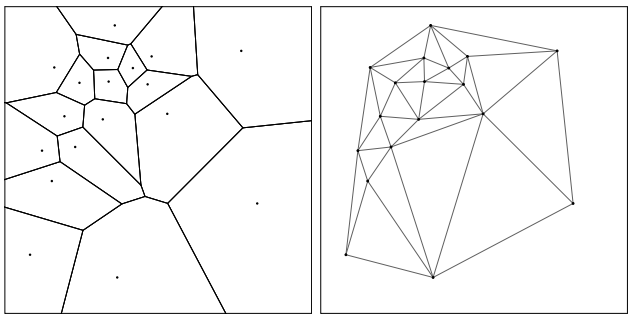


Figure 2 The Voronoi diagram and its dual Delaunay triangulation.

Let P be a set of points in $\mathbb{R}^d$. The *Voronoi cell* of a point $q \in P$ is defined as

$$\text{Vor}_P(q) := \{x \in \mathbb{R}^d \mid \|x - q\| = \mathbf{d}(x, P)\}.$$

For a subset of points $S \subseteq P$, we can define its Voronoi cell as

$$\mathrm{Vor}_P(S) := \bigcap_{q \in S} \mathrm{Vor}_P(q).$$

The same definitions apply to weighted points. For a given α, the Voronoi cell of q is

$$\mathrm{Vor}_{\hat{P}(\alpha)}(q) := \{x \in \mathbb{R}^d \mid \pi_q(x) = \pi(x, \hat{P}(\alpha))\}.$$

It is well-known that the Voronoi cells are polyhedra. The *Voronoi diagram* of P is defined as the polyhedral complex composed of nonempty Voronoi cells $\mathrm{Vor}_P(S)$ for all $S \subseteq P$.

The *Delaunay complex* (also known with some nuances as the Delaunay triangulation, tesselation, or mosaic) is the simplicial complex formed by the subsets $S \subseteq P$ for which $\mathrm{Vor}_P(S)$ is nonempty. The subsets are the *simplices* and the *dimension* of S is $|S| - 1$. We are defining the Delaunay complex here as an abstract simplicial complex. In the special case where all simplices have dimension at most d, the Delaunay complex will embed neatly into $\mathbb{R}^d$ with the vertices embedded at the points of P and each simplex embedded as the convex closure of its vertices. For the purposes of this paper we will not need the embedding and thus will have no need for the usual general position conditions as would usually be required for the geometric realization of the complex in $\mathbb{R}^d$. In fact, we will explicitly construct "degenerate" Delaunay complexes because the adjustment of weights over time will necessarily pass through instants where higher dimensional simplices are present in the Delaunay complex. These are the moments when a flip occurs. We will only require that at most one flip occurs at a time.

2.6 The Kinetic View of Flips

Kinetic data structures [22] generalize the classic sweepline approach of Bentley and Ottmann. The goal is to maintain some geometric structure as points move along trajectories. The principal technique is to rewrite the geometric predicates defining the structure as functions (usually polynomials) of time and then solving (finding roots) for the time when the predicate will no longer hold. At that time, some combinatorial change is required. These changes are stored in a priority queue, ordered by time.

Incremental Delaunay triangulation can be phrased as a kinetic data structures problem if one understand the motion in $d+1$ dimensions as a continuous change in the weight of the point being inserted. This perspective is often abandoned because the precise ordering of the flips is rarely important and is not necessary for correct computation (see Edelsbrunner [20]). However, in our case, we want the precise order and time of the flips that occur, because these inform the final filtration. Also, we will be adding multiple points at once, so the order is important.

A similar approach was used by Miller and Sheehy [30] in an output-sensitive algorithm for computing Delaunay triangulations. In that paper, it was observed that the predicate polynomials are linear for the case where the points are partitioned into two sets, one with weight zero and one with squared weight varying linearly in time.

2.7 Clipping the Voronoi Diagram

The *clipped Voronoi cell* is the intersection of a Voronoi cell and a ball. Let $\hat{P}$ be a weighted point set with weights varying in time as above. Then, for each $p_i \in \hat{P}$ and each $\alpha \geq 0$ there is a ball

$$b_i(\alpha) := \mathrm{ball}(p_i, \min\{\lambda_i, \alpha\}).$$

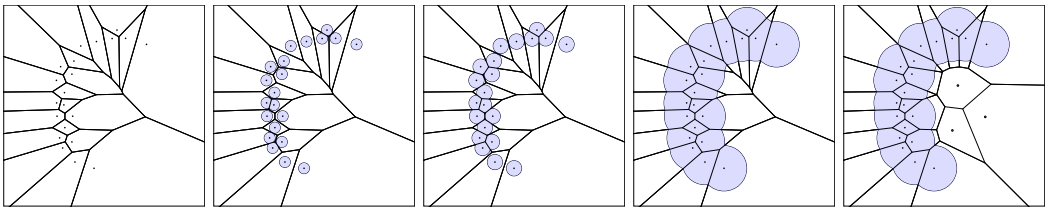


Figure 3 The clipped Voronoi cells exactly cover the offsets. In the last frame, we illustrate how the addition of extra points may not change the offsets or the Delaunay filtration. These extra points are used in Section 4 to keep the complexity linear.

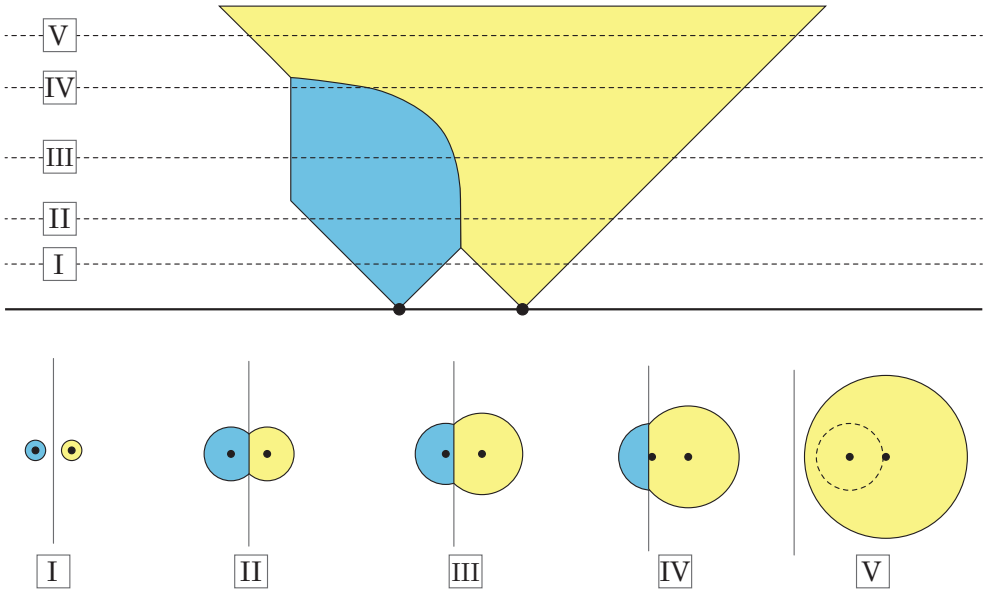


Figure 4 For two points, the lifted, clipped Voronoi cells are shown from the side. The clipped Voronoi diagrams at five different scales are illustrated. Note that at some scale, the left ball ceases growing. Then, it is overtaken by the cell of the right point.

Specifically, for $p_i \in P$, we define the clipped Voronoi cell of p_i as

$$V_i^\alpha := \mathrm{Vor}_{\hat{P}(\alpha)}(p_i) \cap b_i(\alpha).$$

The *Delaunay complex at scale* α is the subcomplex of the Delaunay complex defined by using the clipped Voronoi cells instead of the full Voronoi cells. That is,

$$D^\alpha := \{\sigma \subseteq P \mid \bigcap_{p_i \in \sigma} V_i^\alpha \neq \emptyset\}.$$

By defining the weights as above, we guarantee that for all scales $\alpha \geq (1+\varepsilon)\lambda_k$, the clipped Voronoi cell V_k^α will be empty. This is a direct consequence of Lemma 1, but we give the formal statement below and the proof in the full version for completeness. (See Fig. 4.)

▶ **Lemma 3.** *Let $P \subset \mathbb{R}^d$ be ordered according to a greedy permutation with insertion radii $r_0 \ldots r_{i-1}$. Let $\varepsilon > 0$ be any constant. For all $j \in [1:n]$, let the freezing times λ_j be chosen so that $\lambda_j \geq \frac{1+\varepsilon}{\varepsilon} r_j$. Then, for all $k \in [1:n]$ and all $\alpha > (1+\varepsilon)\lambda_k$, we have $V_k^\alpha = \emptyset$.*

This is how some points will cease to impact the filtration at larger scales. In the next section, we will see how to simulate the removal of vertices whose clipped Voronoi cells are empty. Before we give that construction, we will relate the clipped Voronoi diagram to the (weighted) offsets. For completeness, we include a proof of this well-known fact (see for example Edelsbrunner [15]) in the full version.

▶ **Lemma 4.** *Let $P \subset \mathbb{R}^d$ be ordered according to a greedy permutation with insertion radii $r_0 \dots r_{i-1}$. Let $\varepsilon > 0$ be any constant. For all $j \in [1:n]$, let the freezing times λ_j be chosen so that*

$$\lambda_j \geq \frac{1+\varepsilon}{\varepsilon} r_j.$$

Then, for all $\alpha \geq 0$,

$$\hat{P}^\alpha = \bigcup_{i \in [n]} V_i^\alpha.$$

2.8 Extending Voronoi and Delaunay in Time and Space

Given that we will be considering weighted point sets in which the weights of the points vary in time, it is useful to give a concrete geometric structure that captures the evolution of the Voronoi diagram and the Delaunay triangulation.

Define

$$\hat{P}_+^\alpha := \{(x, \gamma) \in \mathbb{R}^d \times [0, \alpha] \mid x \in \hat{P}^\gamma\}.$$

This filtration has the property that for any $\alpha \geq 0$, there is a natural homotopy equivalence $m : \hat{P}_+^\alpha \to \hat{P}^\alpha$ defined by the projection $m(x, \gamma) = x$. This follows from the fact that $\hat{P}^\alpha$ only grows with increasing α and thus the fibers of m are simply connected (line segments).

We can similarly embed the clipped Voronoi cells in $\mathbb{R}^{d+1}$ by defining

$$V_{+i}^\alpha := \{(x, \gamma) \in \mathbb{R}^d \times [0, \alpha] \mid x \in V_i^\gamma\}.$$

The collection of these sets is

$$V_+^\alpha := \{V_{+i}^\alpha\}_{i \in [n]}.$$

The extended Delaunay complex is defined as

$$D_+^\alpha := \{\sigma \subseteq P \mid \bigcap_{p_i \in \sigma} V_{+i}^\alpha \neq \emptyset\}.$$

So, D_+^α contains every simplex that appears in any Delaunay complex D^γ for any $\gamma \in [0, \alpha]$.

2.9 Nerves

Construction of the Delaunay triangulation from the Voronoi cells is an example of a *nerve*. More generally, given a collection U of sets, we define a simplicial complex

$$\mathrm{Nrv}(U) := \{\sigma \subset U \mid \bigcap_{S \in \sigma} S \neq \emptyset\}.$$

For the Delaunay triangulation, if there is a nonempty intersection of Voronoi cells, we identify the corresponding simplex with the set of points defining those cells. So, D^α is (isomorphic to) the nerve of V^α and D_+^α is the nerve of V_+^α.

The collection U is called a *cover*, and the set $W = \bigcup_{S \in U} S$ is the set that U covers. A cover is a *good cover* if the intersections of elements are all either empty or contractible. In the case of the Voronoi diagram as well as the clipped Voronoi diagram, the convexity of the cells guarantees that the cover is good.

The *Nerve Theorem* says that the nerve of a good cover is homotopy equivalent to the union, so the homology of the nerve matches the homology of the union. In the case of the clipped Voronoi diagram, the Nerve Theorem implies that the Delaunay complex at scale α is equivalent in homology to the weighted offsets. This observation was one of the main ideas that drove the development of persistent homology.

If, instead of a collection of sets, we have a collection of filtrations, we can define their nerve as a filtered simplicial complex. For example, if we have filtrations $\{(U_0^\alpha), \ldots, (U_{k-1}^\alpha)\}$, then for each α we can define the cover $U^\alpha = \{U_0^\alpha, \ldots, U_{k-1}^\alpha\}$ of $W^\alpha = \bigcup_{i \in [k]} U_i^\alpha$ and the nerve $N^\alpha = \mathrm{Nrv}(U^\alpha)$. Then, we have a new filtration (W^α) and a filtered simplicial complex (N^α). The Persistent Nerve Lemma [8] implies that if U^α is a good cover for all α, then the persistence diagrams of (N^α) and (W^α) are identical.

3 The Sparse Delaunay Filtration

In this section, we will prove that our sparse Delaunay filtration $\mathrm{Dgm}(D_+^\alpha)$ is a good approximation to $\mathrm{Dgm}(P^\alpha)$. In Section 2, we established the geometric lemmas that will allow us to relate $\mathrm{Dgm}(P^\alpha)$ with $\mathrm{Dgm}(\bigcup V_+^\alpha)$. We will use the Persistent Nerve Lemma to show that $\mathrm{Dgm}(\bigcup V_+^\alpha) = \mathrm{Dgm}(D_+^\alpha)$, but it will require showing that lifted clipped Voronoi cells form a good cover. Although the slices of these cells are convex at any fixed α, they are not themselves convex as can be seen in the example in Fig. 4. In this section, we will show despite having non-convex sets, V_+^α *is* a good filtered cover. The proof will depend on our careful choice of weights.

3.1 Adding Points in Waves

To define the weight functions, it suffices to establish the freezing times. The specific choice of the freezing times impacts the size (larger freezing times yields a larger filtration), but also the correctness (the cover must be good). For many of the preceding lemmas, it was necessary to choose freezing times so that

$$\lambda_i \geq \frac{1+\varepsilon}{\varepsilon} r_i,$$

where r_i is the insertion radius of p_i in a greedy ordering of the input set P. Recall that ε is the user chosen parameter that will define the accuracy of our approximation.

We will satisfy this requirement by grouping points according to their insertion radius, rounding up to the nearest power of $1+\varepsilon$. All the points in a group will have the same freezing time. Formally, we set

$$\lambda_i := (1+\varepsilon)^{\lceil \log_{1+\varepsilon}(\frac{1+\varepsilon}{\varepsilon} r_i) \rceil}.$$

Rounding the weights simplifies the proofs, but it is an open question as to whether it is necessary for the correctness of the construction.

The filtration (D_+^α) defined with these weight functions on a greedy permutation is *The Sparse Delaunay Filtration.*

3.2 Monotonicity

The first step in showing that V_+^α is a good filtered cover is to show that every Delaunay simplex in D^α appears and disappears at most once. There are two ways that a simplex can be removed at time α. First, it may be that its Voronoi cell becomes empty and therefore it is removed from the Delaunay triangulation entirely. This is the standard case to analyze in flip-based Delaunay computation. Second, it may be that only the clipped Voronoi cell becomes empty. In this case, we must show that it remains empty for the rest of the filtration.

The challenge is that the lifted Voronoi cells are not convex. Aurenhammer et al. [1] showed that this monotonicity does not hold for a related set of weight functions. That paper addressed the problem of computing the Voronoi diagram of parallel halflines and showed that the slices perpendicular to the halflines are weighted Voronoi diagrams. Similar to the current paper, they construct a $d+1$-dimensional decomposition by sweeping through d-dimensional slices. That paper gives an example attributed to Peter Widmayer's research group in which a particular triangle would be flipped out and later flipped back in. Such a non-monotone example highlights the need to be careful in choosing the weights.

▶ **Lemma 5.** *If $\sigma \in D^\alpha$ and $\sigma \in D^\beta$, then $\sigma \in D^\gamma$ for all $\gamma \in [\alpha, \beta]$.*

Proof. Let $\lambda_\sigma = \min_{p_i \in \sigma} \lambda_i$. Lemma 3 implies that no point p_i appears in D^γ for $\gamma > (1+\varepsilon)\lambda_i$. So, it must be that $\beta \leq (1+\varepsilon)\lambda_\sigma$. If $\alpha < \lambda_\sigma$, the points p_i of σ all have weight $w_i(\gamma) = 0$ for all $\gamma \in [\alpha, \lambda_\sigma]$. In that case, the nonempty set $\bigcap_{p_i \in \sigma} V_i^\gamma$ is a subset of $\bigcap_{p_i \in \sigma} V_i^{\lambda_\sigma}$ and so, $\sigma \in D^\gamma$. So, it will suffice to prove the lemma for the case where α and β are in the interval $[\lambda_\sigma, (1+\varepsilon)\lambda_\sigma]$.

For any $\gamma \in [\alpha, \beta]$, let $t = \frac{\gamma^2 - \alpha^2}{\beta^2 - \alpha^2}$.

This choice implies that $\gamma = \sqrt{(1-t)\alpha^2 + t\beta^2}$.

By rounding the freezing times to powers of $(1+\varepsilon)$, there are no freezing times in the open interval (α, β). In particular, $w_i(\alpha) = 0$ iff $w_i(\beta) = 0$. So, for all $p_i \in P$, we have

$$w_i(\gamma)^2 = (1-t)w_i(\alpha)^2 + tw_i(\beta)^2.$$

Let x be a point in the intersection $\bigcap_{p_i \in \sigma} V_i^\alpha$ and let y be a point in $\bigcap_{p_i \in \sigma} V_i^\beta$. These points witness the existence of σ in D^α and D^β respectively. We will show that $z = (1-t)x + ty$ is in $\bigcap_{p_i \in \sigma} V_i^\gamma$.

Let $p_i \in \sigma$ be chosen arbitrarily. Using the convexity of the squared distance to a point,

$$\begin{aligned}
\pi_{i,\gamma}(z)^2 &= \|p_i - z\|^2 + w_i(\gamma)^2 \\
&= \|p_i - z\|^2 + (1-t)w_i(\alpha)^2 + tw_i(\beta)^2 \\
&\leq (1-t)\|p_i - x\|^2 + t\|p_i - y\| + (1-t)w_i(\alpha)^2 + tw_i(\beta)^2 \\
&= (1-t)\pi_{i,\alpha}(x) + t\pi_{i_\beta}(y) \\
&\leq (1-t)\alpha^2 + t\beta^2 \\
&= \gamma.
\end{aligned}$$

So, $z \in b_i(\gamma)$.

Let $p_j \in P$ be any point. Because $x \in \mathrm{Vor}_{\hat{P}(\alpha)}(\sigma)$ and $y \in \mathrm{Vor}_{\hat{P}(\beta)}(\sigma)$, we know that

$$\pi_{j,\alpha}(x)^2 \geq \pi_{i,\alpha}(x)^2, \text{ and } \pi_{j,\beta}(y)^2 \geq \pi_{i,\beta}(y)^2.$$

So, by the definition of the power distance,

$$\begin{aligned}\pi_{j,\gamma}(z)^2 - \pi_{i,\gamma}(z)^2 &= \|p_j - z\|^2 - \|p_i - z\|^2 + w_j(\gamma)^2 - w_i(\gamma)^2 \\ &= \|p_j\|^2 - \|p_i\|^2 - 2z^\top(p_j - p_i) + w_j(\gamma)^2 - w_i(\gamma)^2 \\ &= (1-t)(\pi_{j,\alpha}(x) - \pi_{i,\alpha}(x)) + t(\pi_{j,\beta}(y) - \pi_{i,\beta}(y)) \\ &\geq 0.\end{aligned}$$

So, for all $p_i \in \sigma$ and $p_j \in P$, we have $\pi_{i,\gamma}(z) \leq \pi_{j,\gamma}(z)$ and thus $z \in \mathrm{Vor}_{\hat{P}(\gamma)}(p_i)$. We have shown $z \in b_i^\gamma$ and $z \in \mathrm{Vor}_{\hat{P}(\gamma)}(p_i)$, and therefore, $z \in b_i^\gamma \cap \mathrm{Vor}_{\hat{P}(\gamma)}(p_i) = V_i^\gamma$. As this holds for all $p_i \in \sigma$, we have a point $z \in \bigcap_{p_i \in \sigma} V_i^\gamma$, and thus, $\sigma \in D^\gamma$ as desired. ◀

3.3 A Good Filtered Cover

We can now prove that the lifted Voronoi cells form a good filtered cover.

▶ **Lemma 6.** *Let $\varepsilon > 0$ be any constant. Let V_+^α be the lifted Voronoi cells whose nerve is the ε-Sparse Delaunay Filtration for $P \subset \mathbb{R}^d$. Then, (V_+^α) is a good filtered cover of $(\hat{P}^\alpha)$.*

Proof. Fix any $\alpha \geq 0$. Let $\sigma \in \mathrm{Nrv}(V_+^\alpha)$ be any simplex. We will show that the intersection of the lifted, clipped Voronoi cells $\{V_i^\alpha \mid p_i \in \sigma\}$ is contractible.

In each d-dimensional slice, the clipped Voronoi cells V_i^α are convex, so their intersection is convex. We can deformation retract the nonempty clipped Voronoi cells in each slice to the orthocenter, i.e., the point in the cell that minimizes the maximum weighted distance to the points of σ. The cells change continuously in time and so does the orthocenter of the simplex. The retractions in each slice will be continuous as a retraction in $\mathbb{R}^{d+1}$. By Lemma 5 the collection of orthocenters in the slices will form a connected path and thus are contractible. ◀

▶ **Theorem 7.** *Let $\varepsilon > 0$ be any constant. Let (D_+^α) be the ε-Sparse Delaunay Filtration for $P \subset \mathbb{R}^d$. Then, $\mathrm{Dgm}(D_+^\alpha)$ is a $(1+\varepsilon)$-approximation to $\mathrm{Dgm}(P^\alpha)$.*

Proof. By Lemma 6, the lifted clipped Voronoi cells $\{V_i^\alpha\}$ form a good cover of $\hat{P}^\alpha$ for all α and therefore, by the Persistent Nerve Lemma,

$$\mathrm{Dgm}(D^\alpha) = \mathrm{Dgm}(\hat{P}^\alpha).$$

Lemma 2 gives an interleaving of persistence modules (see [7]) so that $\mathrm{Dgm}(\hat{P}^\alpha)$ is a $(1+\varepsilon)$-approximation to $\mathrm{Dgm}(P^\alpha)$. Combining these facts, we get that $\mathrm{Dgm}(D^\alpha)$ is a $(1+\varepsilon)$-approximation to $\mathrm{Dgm}(P^\alpha)$. ◀

3.4 Size Analysis

In general, the Delaunay complex on n points (in general position) may have $O(n^{\lceil d/2 \rceil})$ simplices [32]. There are several special cases where it is known that the Delaunay complex has size $O(n)$. Most such cases are based on input models that guarantee the points are spaced according to some Poisson process. The analysis invariably depends on showing that the complex is everywhere locally sparse in the sense that every vertex participates in at most a constant number of simplices. It will not be hard to show that a similar bound holds for every slice and, in particular, every slice has linear size (Lemma 8). Then, we will show that the total number of simplices (the union of all slices) also has linear size (Theorem 9).

▶ **Lemma 8.** *For all $\alpha \geq 0$, every vertex in the weighted Delaunay complex D^α has at most $O\left(\left(\frac{1+\varepsilon}{\varepsilon}\right)^d\right)$ neighbors.*

Proof. Let p be any point in P. Let $Q = \{q_0, \ldots, q_{k-1}\}$ be the neighbors of p in D^α. Every edge of D^α has length at most 2α. By Lemma 3, every q_i must have a freezing time at least $\frac{\alpha}{1+\varepsilon}$ and thus, an insertion radius of at least $\frac{\varepsilon\alpha}{(1+\varepsilon)^3}$. So, the points of Q are contained in $\text{ball}(p, 2\alpha)$ and are all pairwise $\frac{\varepsilon\alpha}{(1+\varepsilon)^3}$-separated. By comparing the volumes of the k disjoint empty balls of radius $\frac{\varepsilon\alpha}{2(1+\varepsilon)^3}$ around the points of Q to the volume of the ball of radius that $\left(2 + \frac{\varepsilon}{(1+\varepsilon)^3}\right)\alpha$ that contains them, we get that $k = O\left(\left(\frac{1+\varepsilon}{\varepsilon}\right)^d\right)$. ◀

▶ **Theorem 9.** *Let P be a set of n points in $\mathbb{R}^d$ with greedy weights. Let $\varepsilon \geq 0$ be a constant. The total size of (D^α) is $O(n)$.*

Proof. The proof follows the exact pattern of previous work on sparse filtrations (see [35, 6] for more a more detailed analysis). Each simplex of the filtration is charged to the vertex with the smallest insertion radius. By a packing volume argument analogous to that in Lemma 8, we see that no vertex is charged for more than a constant number of simplices in the final filtration. Thus the total size is $O(n)$ as desired. ◀

4 Efficient Construction

The main challenge to efficiently constructing the Sparse Delaunay Filtration is to avoid constructing the entire Delaunay complex. Doing so could easily negate any efficiency gains from sparsification. In this section, we will describe an approach based on Voronoi refinement that uses extra points called Steiner points to keep the complexity of the Delaunay complex linear in the number of points. None of the Steiner points will appear in the output filtration. They serve only to fill in large gaps that could potentially create a superlinear number of Delaunay simplices.

Earlier work in approximating the persistence diagram of the distance to Euclidean points also used Steiner points [25, 33], but in that case, the Steiner points were an essential part of the filtration. Steiner points and Voronoi refinement have also been used in output-sensitive algorithms to construct D^α [36]. In that case, the output could still be superlinear in the input size depending on the arrangement of the points and the choice of α.

Flip-based algorithms for computing the Delaunay triangulation start the insertion of a new point by flipping a new vertex in with a $(1, d+1)$-flip. This requires that the new point is contained in one of the simplices of the current triangulation. If we ignore or discard simplices in the Delaunay triangulation that do not appear in the subcomplex D^α, then we cannot necessarily flip in new points, because we could have discarded a simplex containing the new point. If we maintain the full Delaunay triangulation at all times, we might store too many simplices. To balance between the two, we use Steiner point to give a sparse representation of the regions far from the input points at a given scale. In Section 4.1, we explain how these Steiner points are chosen. Then, in Section 4.2, we show why these Steiner points do not affect the output. The full algorithm is then presented in Section 4.3, and a detailed analysis is available in the full version.

4.1 Voronoi Refinement

Voronoi cells are polyhedra whose vertices we will call *corners* to distinguish them from the input vertices. We will assume that the affine closure of the points is d-dimensional, so every Voronoi cell has at least one corner. Let $P \subset \mathbb{R}^d$ and let $y \in P$ be any point. Let z be the nearest neighbor of y in P, and let x be the corner of $\text{Vor}_P(y)$ farthest from y. The *aspect* of $\text{Vor}_P(y)$ is

$$\text{aspect}(y) := \frac{\|y - x\|}{\|y - z\|}.$$

The point set P is *τ-well-spaced* if $\text{aspect}(y) < \tau$ for all $y \in P$. There are many advantages to well-spaced points when constructing Delaunay triangulations. A major advantage is that the number of simplices incident to any vertex will be at most a constant. So, the total complexity of the Delaunay triangulation of n well-spaced points is at most $O(n)$.

Voronoi refinement is a variant of Delaunay refinement [10, 31] and is commonly used in mesh generation (see the books by Edelsbrunner [16] and Cheng et al. [9] for more details). The corners of a Voronoi cell are the circumcenters of their dual Delaunay simplices. The basic Voronoi refinement algorithm is to add the farthest corner of any cell $\text{Vor}_P(y)$ for which $\text{aspect}(y) > \tau$. Repeating this process eventually produces a τ-well-spaced set of points. Moreover, the total number of points in the output is asymptotically optimal [31, 34], i.e., the size is within a constant factor of any τ-well-spaced superset of P.

4.2 Why the Steiner points don't appear in the filtration

Every edge in D^α is induced by the intersection of two clipped Voronoi cells, so the length of every edge is at most 2α. The following lemma is the key to guarantee that the Sparse Delaunay Filtration we construct contains no Steiner points as vertices. It shows that the Steiner points are always more than 2α away from any other points, and therefore, there can be no edges incident to a Steiner point in D^α.

▶ **Lemma 10.** *Let ε and α be nonnegative real numbers. Let P and S be subsets of $\mathbb{R}^d$ such that no point of S is within 2α and no point of P is within $\varepsilon\alpha$ of any other point of $P \cup S$. If S' is formed by adding Steiner points to S at the far corners of Voronoi cells with aspect greater than $\frac{2}{\varepsilon}$, then no point of S' will be within 2α of any other point of $P \cup S'$.*

Proof. It suffices to show that the spacing condition holds after the insertion of each Steiner point x. Let y be the point whose Voronoi cell was refined by the addition of x and let z be the nearest neighbor of y. Then, $\|y - z\| \geq \varepsilon\alpha$ and the aspect of the cell is bounded as

$$\frac{2}{\varepsilon} < \frac{\|y - x\|}{\|y - z\|} \leq \frac{\|y - x\|}{\varepsilon\alpha}.$$

It follows that $\|y - x\| > 2\alpha$. Because x was in the Voronoi cell of y, it follows that the distance to any other point is also greater than 2α. ◀

4.3 The Full Algorithm

In the preprocessing phase of the algorithm, we compute a greedy permutation of P. During this computation, we also compute for each p_i, the nearest predecessor in the ordering as well as its distance, the insertion radius r_i. The radius will be used to establish the weights and define the waves. The nearest predecessor will be used for point location when inserting new points.

The algorithm then proceeds by constructing the filtration one wave at a time in order of the greedy permutation. That is, for wave w, the filtration is constructed for the interval $[(1+\varepsilon)^w, (1+\varepsilon)^{w+1}]$. Moreover, as the construction increases the density of points as it goes, it also decreases the radius, so the simplices of the filtration are discovered in reverse order.

At the start of each wave, some set U of points have already been inserted, and some set F of points will be inserted into to the Delaunay triangulation. The points U are unfrozen throughout the wave so their weights will always be zero. The points F are frozen at the start of the wave interval, so their weights will vary equally in time. We start the wave by locating the Delaunay simplices containing each of the points of F. Each pair of a point and the d-simplex that contains it forms a $(d+1)$-simplex. We compute the flip time for each of these simplices and store the flip in the event queue.

Processing the flips only requires that we remove the next flip from the event queue. We check that the simplices are still present in the complex, i.e., that no other flip removed some of its subsimplices. Then, we execute the flip, updating the Delaunay triangulation, and relocating uninserted points of F that were in the removed simplices.

For each flip, we update the birth times of all simplices that may have been affected. That is, if S is the set of $d+2$ points involved in the flip at scale α, we will process each simplex $\sigma \subseteq S$ starting with the highest dimensions. The tentative birth time of every simplex is computed assuming that the structure of the triangulation will not change. A simplex is *finalized* when we add it to the filtration. A simplex is *discarded* if we have established that it will never appear in the filtration. Discarded simplices do not need to be updated in this step. For $\sigma = S$, set $\text{birth}(S) = \alpha$ and either finalize it if $\text{radius}(S) \leq \alpha$, or discard it otherwise. For simplices σ removed by the flip, either finalize it if $\text{birth}(\sigma) \geq \alpha$ or discard it otherwise. For all other simplices, if the newly computed birth time is at most α, then update $\text{birth}(\sigma)$ and finalize σ otherwise.

At the end of each wave, we perform a Voronoi refinement step, adding Steiner points until the points are $\frac{2}{\varepsilon}$-well-spaced. That is, while any Voronoi cell has aspect greater than $\frac{2}{\varepsilon}$, we add its farthest corner. As a consequence of Lemma 10, no changes to the filtration are made at this time.

The overall running time is just the cost of computing a greedy permutation, doing an incremental Delaunay triangulation with sparse refinement, and performing a constant amount of extra work per flip. These are relatively standard analyses, so in the interest of space, they have been relegated to the full version.

5 Conclusion

We have presented a linear size Delaunay filtration for n points in $\mathbb{R}^d$ as well an efficient algorithm to compute it.

It is also relevant to note that this algorithm also can be used to compute a well-spaced set of points. That is, if one keeps the final Delaunay triangulation of the input plus the Steiner points, the result will be well-spaced. Performing a more aggressive Voronoi refinement to achieve a better spacing constant then resembles a standard Voronoi/Delaunay refinement starting from a well-spaced point set. This is substantially simpler than previous algorithms to do Sparse Voronoi Refinement [24, 29] because it obviates any need to "snap" Steiner points to nearby input points. It is not obvious whether the tradeoff between a size increase from the difference in the spacing constant offsets the improvements from simplified point location.

References

1 Franz Aurenhammer, Bert Jüttler, and Günter Paulini. Voronoi diagrams for parallel halflines and line segments in space. In Yoshio Okamoto and Takeshi Tokuyama, editors, *28th International Symposium on Algorithms and Computation (ISAAC 2017)*, volume 92 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 7:1–7:10. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017.

2 Magnus Bakke Botnan and Gard Spreemann. Approximating persistent homology in Euclidean space through collapses. *Applicable Algebra in Engineering, Communication and Computing*, 26(1):73–101, 2015. `arXiv:1403.0533`.

3 A. Bowyer. Computing dirichlet tessellations. *The Computer Journal*, 2(24):162–166, 1981.

4 Kevin Q. Brown. Voronoi diagrams from convex hulls. *Information Processing Letters*, 9(5):223–228, 1979.

5 Mickaël Buchet, Frédéric Chazal, Steve Y. Oudot, and Donald R. Sheehy. Efficient and robust persistent homology for measures. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 168–180, 2015.

6 Nicholas J. Cavanna, Mahmoodreza Jahanseir, and Donald R. Sheehy. A geometric perspective on sparse filtrations. In *Proceedings of the Canadian Conference on Computational Geometry*, 2015.

7 Frédéric Chazal, Vin de Silva, Marc Glisse, and Steve Oudot. *The Structure and Stability of Persistence Modules*. SpringerBriefs in Mathematics. Springer International Publishing, 2016.

8 Frédéric Chazal and Steve Y. Oudot. Towards persistence-based reconstruction in Euclidean spaces. In *Proceedings of the 24th ACM Symposium on Computational Geometry*, pages 232–241, 2008.

9 Siu-Wing Cheng, Tamal K. Dey, and Jonathan Richard Shewchuk. *Delaunay Mesh Generation*. CRC Press, 2012.

10 L. Paul Chew. Guaranteed-Quality Mesh Generation for Curved Surfaces. In *Proceedings of the Ninth Annual Symposium on Computational Geometry*, pages 274–280, 1993.

11 Aruni Choudhary, Michael Kerber, and Sharath Raghvendra. Improved topological approximations by digitization. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2675–2688, 2019.

12 Kenneth L. Clarkson. Nearest neighbor searching in metric spaces: Experimental results for 'sb(s)'. Preliminary version presented at ALENEX99, 2003.

13 Tamal K. Dey, Dayu Shi, and Yusu Wang. Simba: An efficient tool for approximating rips-filtration persistence via simplicial batch-collapse. In *24th Annual European Symposium on Algorithms*, pages 206:1–206:16, 2016.

14 M.E. Dyer and A.M. Frieze. A simple heuristic for the p-centre problem. *Operations Research Letters*, 3(6):285–288, 1985.

15 Herbert Edelsbrunner. The union of balls and its dual shape. *Discrete & Computational Geometry*, 13:415–440, 1995.

16 Herbert Edelsbrunner. *Geometry and Topology for Mesh Generation*. Cambridge University Press, 2001.

17 Herbert Edelsbrunner, David G. Kirkpatrick, and Raimund Seidel. On the shape of a set of points in the plane. *IEEE Transactions on Information Theory*, 29(4):551–559, 1983.

18 Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological persistence and simplification. *Discrete & Computational Geometry*, 4(28):511–533, 2002.

19 Herbert Edelsbrunner and Raimund Seidel. Voronoi diagrams and arrangements. *Discrete & Computational Geometry*, 1(1):25–44, 1986.

20 Herbert Edelsbrunner and Nimish R. Shah. Incremental topological flipping works for regular triangulations. *Algorithmica*, 15, 1996.

21 Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theor. Comput. Sci.*, 38:293–306, 1985.

22 Leonidas J Guibas. Kinetic data structures – a state of the art report. In *Proc. Workshop Algorithmic Found. Robot*, pages 191–209, 1998.

23 Sariel Har-Peled and Manor Mendel. Fast construction of nets in low dimensional metrics, and their applications. *SIAM Journal on Computing*, 35(5):1148–1184, 2006.

24 Benoît Hudson, Gary Miller, and Todd Phillips. Sparse Voronoi Refinement. In *Proceedings of the 15th International Meshing Roundtable*, pages 339–356, Birmingham, Alabama, 2006. Long version available as Carnegie Mellon University Technical Report CMU-CS-06-132.

25 Benoît Hudson, Gary L. Miller, Steve Y. Oudot, and Donald R. Sheehy. Topological inference via meshing. In *Proceedings of the 26th ACM Symposium on Computational Geometry*, pages 277–286, 2010.

26 Barry Joe. Three-dimensional triangulations from local transformations. *SIAM J. Sci. Stat. Comput.*, 10:718–741, 1989.

27 C. L. Lawson. Software for C1 surface interpolation. In J. R. Rice, editor, *Mathematical Software*, volume III, pages 161–194. Academic, New York, 1977.

28 Charles L. Lawson. Transforming triangulations. *Discrete Mathematics*, 3:365–372, 1972.

29 Gary L. Miller, Todd Phillips, and Donald R. Sheehy. Beating the spread: Time-optimal point meshing. In *Proceedings of the 26th ACM Symposium on Computational Geometry*, pages 321–330, 2011.

30 Gary L. Miller and Donald R. Sheehy. A new approach to output-sensitive construction of voronoi diagrams and delaunay triangulations. *Discrete & Computational Geometry*, 52(3):476–491, 2014. doi:10.1007/s00454-014-9629-y.

31 Jim Ruppert. A Delaunay refinement algorithm for quality 2-dimensional mesh generation. *J. Algorithms*, 18(3):548–585, 1995.

32 Raimund Seidel. On the number of faces in higher-dimensional Voronoi diagrams. In *Proceedings of the 3rd Annual Symposium on Computational Geometry*, pages 181–185, 1987.

33 Donald R. Sheehy. *Mesh Generation and Geometric Persistent Homology*. PhD thesis, Carnegie Mellon University, 2011. CMU CS Tech Report CMU-CS-11-121.

34 Donald R. Sheehy. New Bounds on the Size of Optimal Meshes. *Computer Graphics Forum*, 31(5):1627–1635, 2012.

35 Donald R. Sheehy. Linear-size approximations to the Vietoris-Rips filtration. *Discrete & Computational Geometry*, 49(4):778–796, 2013.

36 Donald R. Sheehy. An output-sensitive algorithm for computing weighted α-complexes. In *Proceedings of the Canadian Conference on Computational Geometry*, 2015.

37 D. F. Watson. Computing the n-dimensional delaunay tessellation with application to voronoi polytopes. *The Computer Journal*, 24(2):167–172, 1981.

An Optimal Deterministic Algorithm for Geodesic Farthest-Point Voronoi Diagrams in Simple Polygons

Haitao Wang
Department of Computer Science, Utah State University, Logan, UT, USA

Abstract

Given a set S of m point sites in a simple polygon P of n vertices, we consider the problem of computing the geodesic farthest-point Voronoi diagram for S in P. It is known that the problem has an $\Omega(n + m \log m)$ time lower bound. Previously, a randomized algorithm was proposed [Barba, SoCG 2019] that can solve the problem in $O(n + m \log m)$ expected time. The previous best deterministic algorithms solve the problem in $O(n \log \log n + m \log m)$ time [Oh, Barba, and Ahn, SoCG 2016] or in $O(n + m \log m + m \log^2 n)$ time [Oh and Ahn, SoCG 2017]. In this paper, we present a deterministic algorithm of $O(n + m \log m)$ time, which is optimal. This answers an open question posed by Mitchell in the Handbook of Computational Geometry two decades ago.

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Algorithm design techniques

Keywords and phrases farthest-sites, Voronoi diagrams, triple-point geodesic center, simple polygons

Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.59

Related Version *Full Version*: https://arxiv.org/abs/2103.00076

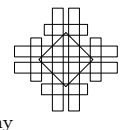
Funding This research was supported in part by NSF under Grant CCF-2005323.

1 Introduction

Let P be a simple polygon of n vertices in the plane. Let S be a set of m points, called *sites*, in P (each site can be either in the interior or on the boundary of P). For any two points in P, their *geodesic distance* is the length of their Euclidean shortest path in P. We consider the problem of computing the geodesic farthest-point Voronoi diagram of S in P, which is to partition P into Voronoi cells such that all points in the same cell have the same farthest site in S with respect to the geodesic distance.

This problem generalizes the Euclidean farthest Voronoi diagram of m sites in the plane, which can be computed in $O(m \log m)$ time [23]; this is optimal as $\Omega(m \log m)$ is a lower bound. For the more general geodesic problem in P, Aronov et al. [3] showed that the complexity of the diagram is $\Theta(n + m)$ and provided an $O(n \log n + m \log m)$ time algorithm. The runtime is close to optimal as $\Omega(n + m \log m)$ is a lower bound. No progress had been made for over two decades until in SoCG 2016 Oh et al. [20] proposed an $O(n \log \log n + m \log m)$ time algorithm. Later in SoCG 2017 Oh and Ahn [19] gave another $O(n + m \log m + m \log^2 n)$ time algorithm and in SoCG 2019 Barba [8] presented a randomized algorithm that can solve the problem in $O(n + m \log m)$ expected time.

In this paper, we give an $O(n + m \log m)$ time deterministic algorithm, which is optimal. The space complexity of the algorithm is $O(n + m)$. This answers an open question posed by Mitchell [17] in the Handbook of Computational Geometry two decades ago.



1.1 Related work

If all sites of S are on the boundary of P, then better results exist. The algorithm of Oh et al. [20] can solve the problem in $O((n+m)\log\log n)$ time while the randomized algorithm of Barba [8] runs in $O(n+m)$ expected time.

The geodesic nearest-point Voronoi diagram for sites in a simple polygon has also attracted much attention. The problem also has an $\Omega(n+m\log m)$ time lower bound. The first close-to-optimal algorithm was given by Aronov [2] and the running time is $O((n+m)\log(n+m)\log n)$. Papadopoulou and Lee [21] improved the algorithm to $O((n+m)\log(n+m))$ time. Recent progress has been made by Oh and Ahn [19] who presented an $O(n+m\log m\log^2 n)$ time algorithm and also by Liu [16] who designed an $O(n+m(\log m+\log^2 n))$ time algorithm. Finally the problem was solved optimally in $O(n+m\log m)$ time by Oh [18].

Another closely related problem is to compute the *geodesic center* of a simple polygon P, which is a point in P that minimizes the maximum geodesic distance from all points of P. Asano and Toussaint [4] first gave an $O(n^4\log n)$ time algorithm for the problem. Pollack et al. [22] derived an $O(n\log n)$ time algorithm. Recently the problem was solved optimally in $O(n)$ time by Ahn et al. [1]. The *geodesic diameter* of P is the largest geodesic distance between any two points in P. Chazelle [9] first gave an $O(n^2)$ time algorithm and then Suri [24] presented an improved $O(n\log n)$ time solution. Hershberger and Suri [13] finally solved the problem in $O(n)$ time.

All above results are for simple polygons. For polygons with holes, the problems become more difficult. The geodesic nearest-point Voronoi diagram for m point sites in a polygon with holes of n vertices can be solved in $O((n+m)\log(n+m))$ time by the algorithm of Hershberber and Suri [14]. Bae and Chwa [5] gave an algorithm for constructing the geodesic farthest-point Voronoi diagram and the algorithm runs in $O(nm\log^2(n+m)\log m)$ time. For computing the geodesic diameter in a polygon with holes, Bae et al. [6] solved the problem in $O(n^{7.73})$ or $O(n^7(h+\log n))$ time, where h is the number of holes. For computing the geodesic center, Bae et al. [7] first gave an $O(n^{12+\epsilon})$ time algorithm, for any constant $\epsilon>0$; Wang [26] solved the problem in $O(n^{11}\log n)$ time.

1.2 Our approach

We follow the algorithm scheme in [19], which follows [3]. Specifically, we first compute the geodesic convex hull of all sites of S in $O(n+m\log m)$ time [11, 12, 25], and then compute the geodesic center c^* of the hull in $O(n+m)$ time [1]. Aronov [3] showed that the farthest Voronoi diagram forms a tree with c^* as the root and all leaves on ∂P, the boundary of P. We construct the farthest Voronoi diagram restricted to ∂P; this can be done in $O(n+m)$ time by a recent algorithm of Oh et al. [20] once the geodesic convex hull of S is known.

Next we run a *reverse geodesic sweeping* algorithm to extend the diagram from ∂P to the interior of P (i.e., based on all leaves on ∂P and the root c^* of the tree, we want to construct the tree). Here we use a *geodesic sweeping circle* that consists of all points with the same geodesic distance from c^*. Aronov [3] implemented this sweeping algorithm in $O((n+m)\log(n+m))$ time. Oh and Ahn [19] gave an improved solution of $O(n+m\log m+m\log^2 n)$ time by using a data structure for the following query problem: Given three points in P, compute the point that is equidistant from them. Oh and Ahn [19] built a data structure in $O(n)$ time that can answer each query in $O(\log^2 n)$ time, and that is why the time complexity of their algorithm has a $\log^2 n$ factor. We improve the query time to $O(\log n)$ (with $O(n)$ time preprocessing) with the help of the following observations. First, the three points involved in a query are three sites of S whose Voronoi cells are adjacent along the sweeping circle. Second, among

the three sites involved in a query, for every two sites whose Voronoi cells are adjacent, the sweeping algorithm provides us with a point equidistant to them. These observations along with the tentative prune-and-search technique of Kirkpatrick and Snoeyink [15] lead us to a query algorithm of $O(\log n)$ time. Consequently, the sweeping algorithm can be implemented in $O(n + m \log m)$ time.

We should point out that in her algorithm for computing the geodesic nearest-point Voronoi diagram, Oh [18] also announced an $O(\log n)$ time algorithm for the above query problem and her algorithm also uses the tentative prune-and-search technique (although the details are omitted due to the page limit). However, the difference is that she uses a balanced geodesic triangulation [10] and her result is based on the assumption that the sought point of the query lies in a known geodesic triangle $\triangle$ and the three query points are in the same subpolygon of P separated by a side of $\triangle$ (see Lemma 4.2 in [18]). For our problem, we do not need the balanced geodesic triangulation and do not have such an assumption. Instead, our algorithm relies on the observations mentioned above.

The rest of the paper is organized as follows. Section 2 defines notation and introduces some concepts. The algorithm for constructing the geodesic Voronoi diagram is described in Section 3. Section 4 presents the algorithm for a lemma about the query problem discussed above. Due to the space limit, some proofs are omitted but can be found in the full paper.

2 Preliminaries

Like the previous work [3, 8, 19, 20], for ease of discussion, we make a general position assumption that no vertex of P is equidistant from two sites of S and no point of P has four farthest sites. We occasionally use *polygon vertex* to refer to a vertex of P and use *polygon edge* to refer to an edge of P.

For any two points p and q in P, let $\pi(p, q)$ denote the (Euclidean) shortest path from p to q in P; let $d(p, q)$ denote the length of $\pi(p, q)$. $\pi(p, q)$ is also called the *geodesic path* and $d(p, q)$ is called the *geodesic distance* between p and q. The vertex of $\pi(p, q)$ adjacent to q (resp., p) is called the *anchor* of q (resp., p) in $\pi(p, q)$.

For any two points a and b in the plane, denote by $\overline{ab}$ the line segment with a and b as endpoints, and denote by $|\overline{ab}|$ the length of the segment.

For two sites s and t of S, their *bisector*, denoted by $B(s, t)$, consists of all points of P equidistant from them, i.e., $B(s, t) = \{p \mid d(s, p) = d(t, p), p \in P\}$. Due to the general position assumption, Aronov et al. [2] showed that $B(s, t)$ is a smooth curve connecting two points on ∂P with no other points common with ∂P and $B(s, t)$ comprises $O(n)$ straight and hyperbolic arcs (a straight arc is a line segment); the endpoints of the arcs are *breakpoints*, each of which is the intersection of $B(s, t)$ and a segment extended from a polygon vertex u to another polygon vertex v such that u is an anchor of v in $\pi(s, v)$ or in $\pi(t, v)$ (it is possible that $u = s$ or $u = t$); e.g., see Fig. 1.

For any site $s \in S$, define $C(s)$ as the region consisting of all points p of P whose farthest site is s, i.e., $C(s) = \{p \mid d(p, s) \geq d(p, s'), s' \in S\}$. We call $C(s)$ the (farthest) *Voronoi cell* of s. Note that $C(s)$ may be empty; if $C(s)$ is not empty, then it is simply connected [3]. The Voronoi cells of all sites of S form a partition of P. We define the *geodesic farthest-point Voronoi diagram* (or *farthest Voronoi diagram* for short), denoted by $FVD(S)$, as the closure of the interior of P minus the union of the interior of $C(s)$ for all $s \in S$; alternatively, $FVD(S) = \{p \in B(s, t) \mid s, t \in S \text{ and } d(s, p) = \max_{r \in S} d(r, p)\}$. A point v of $FVD(S)$ is a *Voronoi vertex* if it is an intersection of a bisector with ∂P or if it has degree 3 (i.e., it has three equidistant sites). The curve of $FVD(S)$ connecting two adjacent vertices is called a

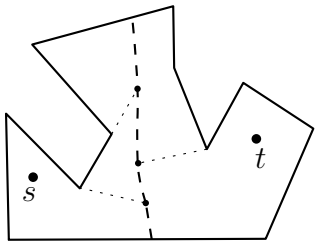


Figure 1 Illustrating the bisector $B(s,t)$ (the dashed curve) with three breakpoints.

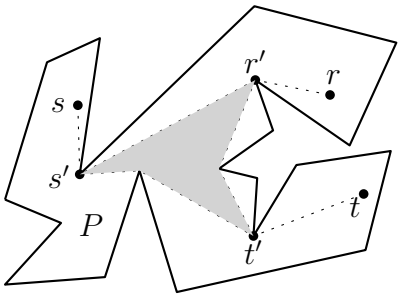


Figure 2 Illustrating a geodesic triangle $\triangle(s,t,r)$ (the gray region).

Voronoi edge, which is a portion of a bisector of two sites. Note that a Voronoi edge may not be of constant size because it may contain multiple breakpoints. While $FVD(S)$ has $O(m)$ Voronoi vertices and edges, the total complexity of $FVD(S)$ is $O(n+m)$ [3].

A subset P' of P is *geodesically convex* if $\pi(p,q)$ is in P' for any two points p and q in P'. The *geodesic convex hull* of S in P, denoted by $GCH(S)$, is the common intersection of all geodesically convex sets containing S. $GCH(S)$ is a weakly simple polygon of at most $n+m$ vertices. Let c^* be the geodesic center of $GCH(S)$, which is also the geodesic center of S [3]. Note that c^* must be on a Voronoi edge of $FVD(S)$. Indeed, if c^* has three farthest sites in S, then c^* is a Voronoi vertex; otherwise it has two farthest sites and thus is in the interior of an edge of $FVD(S)$. Aronov [3] proved that $FVD(S)$ is a tree with c^* as the root and all leaves on ∂P; he also showed that only sites on the boundary of $GCH(s)$ have nonempty cells in $FVD(S)$ and the ordering of the sites with nonempty cells around the boundary of $GCH(s)$ is the same as the ordering of their Voronoi cells around ∂P (the *ordering lemma*). Note that a site on the boundary of $GCH(s)$ may still have an empty Voronoi cell and intuitively this is because P is not large enough [3].

Consider any three points s,t,r in P. The vertex farthest to s in $\pi(s,t) \cap \pi(s,r)$ is called the *junction vertex* of $\pi(s,t)$ and $\pi(s,r)$. The closure of the interior of the geodesic convex hull $GCH(s,t,r)$ is called the *geodesic triangle* of s, t, and r, denoted by $\triangle(s,t,r)$, whose boundary is composed of three convex chains $\pi(s',t')$, $\pi(t',r')$, $\pi(r',s')$, where s' is the junction vertex of $\pi(s,t)$ and $\pi(s,r)$, and t' and r' are defined likewise; e.g., see Fig. 2. The three convex chains are called *sides* of $\triangle(s,t,r)$. The three vertices s', t', and r' are called the *apexes* of $\triangle(s,t,r)$.

3 Computing the farthest Voronoi diagram $FVD(S)$

In this section, we present our algorithm for computing the farthest Voronoi diagram $FVD(S)$.

First, we compute the geodesic convex hull $GCH(S)$ of S in $O(n+m\log m)$ time [11,12,25]. Second, we compute the geodesic center c^* of $GCH(S)$ in $O(n+m)$ time [1]. Third, we compute the portion of $FVD(S)$ restricted to the polygon boundary ∂P, i.e., the leaves of

$FVD(S)$. This can be done in $O(n+m)$ time by the algorithm in [20].[1] The fourth step is to extend the diagram to the interior of P, i.e., construct the tree $FVD(S)$ based on all its leaves and the root c^*. This is achieved by a reverse geodesic sweeping algorithm, whose details are described below.

The algorithm first computes the adjacency information of $FVD(S)$. Specifically, we will compute the locations of all Voronoi vertices of $FVD(S)$; for Voronoi edges, however, we will not compute them exactly (i.e., the locations of their breakpoints will not be computed) but only output their incident Voronoi vertices, i.e., if u and v are two Voronoi vertices incident to the same Voronoi edge, then we will output the pair (u, v) as an *abstract* Voronoi edge. In this way, we will output the abstract tree $FVD(S)$ with the exact locations of all Voronoi vertices; this is called the *topological structure* of $FVD(S)$ in [19]. After having the topological structure, Oh and Ahn [19] gave an algorithm that can construct $FVD(S)$ in additional $O(n + m \log m)$ time. More specifically, with $O(n)$ time preprocessing, each Voronoi edge can be computed in $O(\log n + k)$ time, where k is the number of breakpoints in the Voronoi edge (see Section 4 of [19] for details). As $FVD(S)$ has $O(m)$ Voronoi edges, the total time for computing all Voronoi edges is $O(m \log n + K)$, where K is the total number of breakpoints on all Voronoi edges. As $K = O(n + m)$ [3], the total time is bounded by $O(n + m \log n)$, which is $O(n + m \log m)$.[2] In the following, we will focus on computing the topological structure of $FVD(S)$.

We use a reverse geodesic sweeping as in [3, 19]. Roughly speaking, the sweep line is a *geodesic circle* C consisting of all points in P that have the same geodesic distance from the geodesic center c^* of S. This statement is actually not quite accurate as initially the sweep circle is just the boundary of P. During the sweeping, we maintain the sites whose Voronoi cells currently intersect C; these sites are stored in a cyclic linked list $\mathcal{L}$ ordered by their intersections with C. Initially when $C = \partial P$, as we already have the leaves of $FVD(S)$, we can build $\mathcal{L}$ in $O(n + m)$ time. Note that $|\mathcal{L}| = O(m)$. During the algorithm, C will shrink until c^*; an event happens when C hits a Voronoi vertex, which will be computed on the fly. Specifically, for each triple of adjacent sites s, t, r in the list $\mathcal{L}$, we compute the point, denoted by $\alpha(s, t, r)$, equidistant from them, which is the intersection of the bisectors $B(s, t)$ and $B(t, r)$. Due to our general position assumption, $\alpha(s, t, r)$ is unique if it exists (see Lemma 2.5.3 [3]). We store all these α-points in a priority queue Q, ordered by decreasing geodesic distance from c^*. In order to compute the α-points, for any pair of adjacent sites s and t in $\mathcal{L}$, we maintain a Voronoi vertex, denoted by $\beta(s, t)$, on their bisector $B(s, t)$ with the following property: $\beta(s, t)$ is outside or on the current geodesic circle C. Initially, we set $\beta(s, t)$ to be the Voronoi vertex on ∂P incident to the Voronoi cells of s and t; so the above property holds as $C = \partial P$.

The main loop of the algorithm works as follows. As long as Q is not empty, we repeatedly extract the point with largest geodesic distance from c^* and let the point be $\alpha(s, t, r)$ defined by three sites s, t, r in this order in $\mathcal{L}$. We report $\alpha(s, t, r)$ as a Voronoi vertex and report

[1] Note that the result was not explicitly given in [20] but can be obtained from their $O(n \log\log n + m \log m)$-time algorithm for computing $FVD(S)$. Indeed, given $GCH(S)$, the algorithm first partitions P in $O(n + m)$ time into $O(1)$ subpolygons such that each subpolygon P' is for a problem instance where all involved sites are on the boundary of P' (see Section 7 [20]). Then, each problem instance is further reduced in linear time to a problem instance where all sites are vertices of P' (see Section 6 [20]), and each such problem instance can be solved in linear time (see Section 3 [20]). The total running time of all above is $O(n + m)$ (for computing $FVD(S)$ restricted to the boundary of P only). This result was also used by Oh and Ahn [19] in their $O(n + m \log m + m \log^2 n)$-time algorithm for computing $FVD(S)$.

[2] Indeed, if $m < n/\log n$, then $n + m \log n = \Theta(n)$, which is $O(n + m \log m)$; otherwise, $\log n = O(\log m)$ and $n + m \log n = O(n + m \log m)$.

$(\beta(s,t),\alpha(s,t,r))$ and $(\beta(t,r),\alpha(s,t,r))$ as two abstract Voronoi edges. We remove t from $\mathcal{L}$ and set $\beta(s,r)=\alpha(s,t,r)$. Let x be the neighbor of s other than r in $\mathcal{L}$ and let y be the neighbor of r other than s. We remove $\alpha(x,s,t)$ and $\alpha(t,r,y)$ from Q if they exist. Next, we compute $\alpha(x,s,r)$ and $\alpha(s,r,y)$ (if exist) as well as their geodesic distances from c^*, and insert them into Q.

For the running time, there are $O(m)$ events, for the total number of Voronoi vertices of $FVD(S)$ is $O(m)$ [3], and thus the total time of the algorithm is $O(m\cdot\sigma)$, where $O(\sigma)$ is the time for computing each α point. Lemma 1, which will be proved later in Section 4, is for computing the α-points.

▶ **Lemma 1.** *With $O(n)$ time preprocessing, for any triple of adjacent sites s,t,r in $\mathcal{L}$ at any moment during the algorithm, given the two Voronoi vertices $\beta(s,t)$ and $\beta(t,r)$, our algorithm can do the following in $O(\log n)$ time: if $\alpha(s,t,r)$ is a Voronoi vertex, then compute it; otherwise, either compute $\alpha(s,t,r)$ or return null.*

We remark that Lemma 1 is sufficient for the correctness of our geodesic sweeping algorithm as only Voronoi vertices are essential. If the algorithm returns null, the event will not be inserted to Q. With Lemma 1 at hand, our geodesic sweeping algorithm computes the topological structure of $FVD(S)$ in $O(n+m\log m)$ time. After that, as discussed above, we can compute the full diagram $FVD(S)$ in additional $O(n+m\log m)$ time by the techniques of Oh and Ahn [19]. Also, the space of the algorithm is bounded by $O(n+m)$.

▶ **Theorem 2.** *The geodesic farthest-point Voronoi diagram of a set of m points in a simple polygon of n vertices can be computed in $O(n+m\log m)$ time and $O(n+m)$ space.*

4 Algorithm for Lemma 1

In this section, we present our algorithm for Lemma 1. We first present an algorithm in Section 4.1 for the following *triple-point geodesic center query* problem: given any three points in P, compute their geodesic center in P, which is a point that minimizes the largest geodesic distance from the three query points. Oh and Ahn [19] solved this problem in $O(\log^2 n)$ time, after $O(n)$ time preprocessing. Our algorithm runs in $O(\log n)$ time also with $O(n)$ time preprocessing.[3] This algorithm will be used as a subroutine in our algorithm for Lemma 1, which will be discussed in Section 4.2.

4.1 The triple-point geodesic center query problem

As preprocessing, we construct the two-point shortest path query data structure by Guibas and Hershberger [11, 12] and we refer to it as the *GH data structure*. The data structure can be constructed in $O(n)$ time, after which given any two points p and q in P, the geodesic distance $d(p,q)$ can be computed in $O(\log n)$ time and the geodesic path $\pi(p,q)$ can be output in additional time linear in the number of edges of $\pi(p,q)$.

Consider three query points s, t, and r in P. Our goal is to compute their geodesic center, denoted by c. We follow the algorithm scheme in [19]. Consider the geodesic convex hull $GCH(s,t,r)$ and the geodesic triangle $\triangle(s,t,r)$. We know that c is the geodesic center of $GCH(s,t,r)$ [3]. Depending on whether c is in the interior of $\triangle(s,t,r)$, there are two cases.

[3] To be fair, this problem is not a dominant one in their algorithm, which might be a reason Oh and Ahn [19] did not push their result further.

4.1.1 c is not in the interior of $\triangle(s,t,r)$

If c is not in the interior of $\triangle(s,t,r)$, then it must be on the geodesic path of two points of $\{s,t,r\}$. Without loss of generality, we assume that $c \in \pi(s,t)$. Note that c must be the middle point of $\pi(s,t)$. To locate c in $\pi(s,t)$, we wish to do binary search on the vertices of $\pi(s,t)$. It was claimed in [19] that the query algorithm of the GH data structure returns $\pi(s,t)$ as a binary tree (so that binary search can be done in a straightforward way), in particular, when the simpler approach in [12] is utilized. In fact, this is not quite correct. Indeed, the binary tree structures in both [11] and [12] are used for representing convex chains (or more rigorously, *semiconvex chains* [12]). However, $\pi(s,t)$ is actually a *string* [11], which in general is not a semiconvex chain. The data structure for representing a string is a tree but not a binary tree because a node in the tree may have three children.

For completeness, we provide a binary search scheme on the geodesic path $\pi(p,q)$ returned by the GH data structure for any two query points p and q in P. Suppose we are looking for either a vertex or an edge of $\pi(p,q)$, denoted by w^* in either case, and we have access to an *oracle* such that given any vertex $v \in \pi(p,q)$, the oracle can determine whether w^* is in $\pi(p,v)$ or in $\pi(v,q)$. Then, we have the following lemma (whose proof is omitted).

▶ **Lemma 3.** *With $O(n)$ time preprocessing, given any two query points p and q, the sought vertex or edge w^* can be located by a binary search algorithm that calls the oracle on $O(\log n)$ vertices of $\pi(p,q)$, and the total time of the binary search excluding the time for calling the oracle is $O(\log n)$. In particular, the middle point of $\pi(p,q)$ can be found in $O(\log n)$ time.*

With Lemma 3 at hand, we can find c on $\pi(s,t)$ in $O(\log n)$ time.

We can determine whether c is in the interior of $\triangle(s,t,r)$ in $O(\log n)$ time using Lemma 3 as follows. First, we determine whether c is the middle point of $\pi(s,t)$. To do so, we first compute the middle point p_{st} of $\pi(s,t)$ by Lemma 3. Then, we compute $d(s,p_{st})$ and $d(r,p_{st})$ in $O(\log n)$ time using the GH data structure. It is not difficult to see that p_{st} is c if and only if $d(s,p_{st}) \geq d(r,p_{st})$. If $p_{st} \neq c$, then we use the same way to determine whether the middle point of $\pi(s,r)$ (resp., $\pi(r,t)$) is c. If the above algorithm fails to locate c, then we know that c is in the interior of $\triangle(s,t,r)$.

The above finds c in $O(\log n)$ time for the case where c is not in the interior of $\triangle(s,t,r)$.

4.1.2 c is in the interior of $\triangle(s,t,r)$

We proceed to the case where c is in the interior of $\triangle(s,t,r)$. Our algorithm utilizes the tentative prune-and-search technique of Kirkpatrick and Snoeyink [15].

First observe that in this case c must be equidistant from all three points s, t, and r. Let s' be the junction vertex of $\pi(s,t)$ and $\pi(s,r)$. Define t' and r' similarly. With the GH data structure, each junction vertex can be computed in $O(\log n)$ time [11].

Define p_s, p_t, and p_r to be the anchors of c in $\pi(s,c)$, $\pi(t,c)$, and $\pi(r,c)$, respectively. Note that the segment connecting c to p_s (resp., p_t, p_r) is tangent to the side of $\triangle(s,t,r)$ that contains it. As c is equidistant to s, t, and r, c is the common intersection of the three bisectors $B(s,t)$, $B(s,r)$, and $B(t,r)$. The proof of the lemma below is omitted.

▶ **Lemma 4.** *The middle point p_{st} of $\pi(s,t)$ must be in $\pi(s',t')$; the middle point p_{sr} of $\pi(s,r)$ must be in $\pi(s',r')$; the middle point p_{tr} of $\pi(t,r)$ must be in $\pi(t',r')$.*

Since p_s is the anchor of c in $\pi(s,c)$, p_s has smaller geodesic distance from s than from t or r. Hence, by Lemma 4, p_s must be in $\gamma_s = \pi(s',p_{st}) \cup \pi(s',p_{sr})$, which consists of two convex chains; we call γ_s a *pseudo-convex chain*. Similarly, p_t must be in $\gamma_t = \pi(t',p_{st}) \cup \pi(t',p_{tr})$

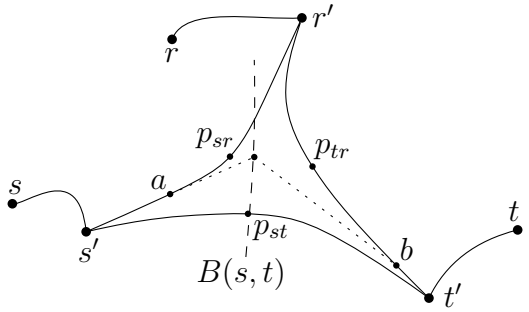


Figure 3 Illustrating the geodesic triangle $\triangle(s,t,r)$ and the function $f(a)$ for $a \in A = \gamma_s$.

and p_r must be in $\gamma_r = \pi(r', p_{tr}) \cup \pi(r', p_{sr})$. We consider p_{st} and p_{sr} as two ends of γ_s. If we move a point p on γ_s from one end to the other, then the slope of the tangent line of γ_s at p continuously changes. Further, p_s is the only point on γ_s such that $\overline{cp_s}$ is tangent to γ_s. Similar properties hold for γ_t, p_t, γ_r, and p_r.

With the above discussion, we now describe our algorithm for computing c. First, we compute the three junction vertices and the three middle points s', t', r', p_{st}, p_{sr}, and p_{tr}. This can be done in $O(\log n)$ time using the GH data structure. To compute c (as well as locate p_s, p_t, and p_r), we resort to the tentative prune-and-search technique [15], as follows.

To avoid the lengthy background explanation, we follow the notation in [15] without definition. We will rely on Theorem 3.9 in [15]. To this end, we need to define three continuous and monotone-decreasing functions f, g, and h. We define them in a way similar to Theorem 4.10 in [15] for finding a point equidistant to three convex polygons. Indeed, our problem may be considered as a weighted case of their problem because each point in our pseudo-convex chains has a weight equal to its geodesic distance from one of s, t, and r.

We parameterize over $[0,1]$ each of the three pseudo-convex chains $A = \gamma_s$, $B = \gamma_t$, and $C = \gamma_r$ from one end to the other in counterclockwise order around $\triangle(s,t,r)$. For example, without loss of generality, we assume that s', t', and r' are counterclockwise around $\triangle(s,t,r)$. Then, γ_s is parameterized from p_{sr} to p_{st} over $[0,1]$, i.e., each value of $[0,1]$ corresponds to a slope of a tangent at a point on γ_s. For each point a of A, we define $f(a)$ to be the parameter of the point $b \in B$ such that the tangent of A at a and the tangent of B at b intersect at a point on the bisector $B(s,t)$ of s and t (e.g., see Fig. 3). Similarly, we define $g(b)$ for $b \in B$ with respect to C and define $h(c)$ for $c \in C$ with respect to A. One can verify that all three functions are continuous and monotone-decreasing (the tangent at an apex of $\triangle(s,t,r)$ is not unique but the issue can be handled [15]). The fixed-point of the composition of the three functions $h \cdot g \cdot f$ corresponds to c, which can be computed by applying the tentative prune-and-search algorithm of Theorem 3.9 [15].

To see that the algorithm can be implemented in $O(\log n)$ time, we need to show that given any $a \in A$ and any $b \in B$, we can determine whether $f(a) > b$ in $O(1)$ time. To this end, we first find the intersection p of the tangent of A at a and the tangent of B at b. Then, $d(s,a) + |\overline{pa}| < d(t,b) + |\overline{pb}|$ if and only if $f(a) > b$. We will discuss below that the values $d(s,a)$ and $d(t,b)$ will be available during the tentative prune-and-search algorithm. Note that here the tangent of A at a actually refers to the half-line of the tangent whose concatenation with $\pi(s',a)$ is still a convex chain (so that the shortest path can follow that half-line), as shown in Fig. 3. Hence, it is possible that the tangent half-line of a does not intersect the tangent half-line of b. If that happens, either the tangent half-line of a intersects the backward extension of the tangent half-line of b or the backward extension of the tangent

half-line of a intersects the tangent half-line of b; in the former case we have $f(a) < b$ and in the latter case $f(a) > b$. Similar properties hold for functions of g and h. Finally, we show that we have appropriate data structures to represent the three pseudo-convex chains A, B, and C so that the algorithm can terminate in $O(\log n)$ rounds. We only discuss A since other two are similar. When the algorithm picks the first vertex of A to test, we will use the vertex s'. After the test, the algorithm will proceed on A on one side of s', say, on $\pi(s', p_{st})$. We apply the binary search scheme of Lemma 3 on $\pi(s', p_{st})$, which will test $O(\log n)$ vertices. Further, whenever a vertex $a \in \pi(s', p_{st})$ is tested, the binary search scheme of Lemma 3 can keep track of $d(s, a)$. Therefore, applying the tentative prune-and-search technique in Theorem 3.9 [15] can compute the geodesic center c in $O(\log n)$ time.

The lemma below summarizes our result on the triple-point geodesic center query problem.

▶ **Lemma 5.** *With $O(n)$ time preprocessing, the geodesic center of any three query points in P can be computed in $O(\log n)$ time.*

4.2 Proving Lemma 1

With Lemma 5, we are ready to present our algorithm for Lemma 1. Consider any three sites s, t, r as specified in the statement of Lemma 1. Our goal is to compute the point $\alpha(s, t, r)$, which is equidistant from the three sites. Recall that we have two points $\beta(s, t)$ and $\beta(t, r)$ available for us, which are critical to the success of our approach.

The first step of our algorithm is to apply the algorithm for Lemma 5 to compute the geodesic center c of the three sites. We check whether c is equidistant to the three sites, in $O(\log n)$ time. If yes, $\alpha(s, t, r) = c$ and we are done. In the following we assume otherwise.

Our algorithm may not compute $\alpha(s, t, r)$ even if it exists, but will guarantee to do so if $\alpha(s, t, r)$ is a Voronoi vertex of $FVD(S)$. This is sufficient for constructing $FVD(S)$ correctly. Hence, in what follows we assume that $\alpha(s, t, r)$ is a Voronoi vertex. This implies that there are two Voronoi edges connecting $\alpha(s, t, r)$ with $\beta(s, t)$ and $\beta(t, r)$, respectively. Recall that c^* is the geodesic center of S. The following lemma was discovered by Aronov et al. [3].

▶ **Lemma 6** (Aronov et al. [3])**.** *If a point p moves from $\beta(s, t)$ (resp., $\beta(t, r)$) to $\alpha(s, t, r)$ along the Voronoi edge, both $d(c^*, p)$ and $d(t, p)$ are monotonically decreasing.*

To simplify the notation, unless otherwise stated, we use α to refer to $\alpha(s, t, r)$. We define the three junction vertices s', t', and r' in the same way as before, which are the three apexes of the geodesic convex hull $\triangle(s, t, r)$. Without loss of generality, we assume that s', t', and r' are counterclockwise around the boundary of $\triangle(s, t, r)$ (e.g., see Fig. 2). We define p_s, p_t, and p_r as the anchors of α in $\pi(s, \alpha)$, $\pi(t, \alpha)$, and $\pi(r, \alpha)$, respectively. Define p_{st}, p_{sr}, and p_{tr} as the middle points of $\pi(s, t)$, $\pi(s, r)$, and $\pi(t, r)$, respectively.

Lemma 7, obtained from the results of Aronov [2], will occasionally be used later.

▶ **Lemma 7** (Aronov [2])**.** *Suppose x and y are two points of P such that their bisector $B(x, y)$ does not contain any vertex of P. Then, for any point $z \in P$, the shortest path $\pi(x, z)$ (resp., $\pi(y, z)$) either does not intersect $B(x, y)$ or intersects it at a single point.*

Recall that $\alpha(s, t, r)$ is not c. Hence, c must be equidistant to two sites and the geodesic distance from them to c is strictly larger than that from the third site to c. Depending on what the two sites are, there are three cases $d(c, s) = d(c, r) > d(c, t)$, $d(c, t) = d(c, r) > d(c, s)$, and $d(c, t) = d(c, s) > d(c, r)$. The following lemma (whose proof is omitted) shows that the latter two cases cannot happen.

▶ **Lemma 8.** *Neither $d(c, t) = d(c, r) > d(c, s)$ nor $d(c, t) = d(c, s) > d(c, r)$ can happen.*

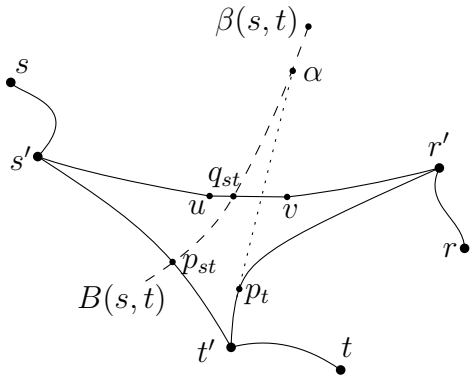


Figure 4 Illustrating the subcase $c \in \pi(s', r')$: c is on $\overline{uv}$.

Note that Lemma 8 is obtained based on the assumption that $\alpha(s,t,r)$ is a Voronoi vertex of $FVD(S)$. Therefore, if one of the two cases in Lemma 8 happens during the algorithm, then we can simply return null.

In what follows, we assume that $d(c,s) = d(c,r) > d(c,t)$. Thus, c must be the middle point of $\pi(s,r)$. Depending on where the location of c is, there are three cases: $c \in \pi(s', r')$, $c \in \pi(s, s') \setminus \{s'\}$, and $c \in \pi(r', r) \setminus \{r'\}$. The latter two cases are symmetric, so we will only discuss the first two cases.

4.2.1 The case $c \in \pi(s', r')$

Let $\overline{uv}$ be the edge of $\pi(s', r')$ containing c such that $d(s,u) < d(s,v)$. It possible that u is s or/and v is t. We first assume that both u and v are polygon vertices; we will show later the other case can be reduced to this case.

Since both u and v are polygon vertices, $\overline{uv}$ divides P into two sub-polygons; one of them, denoted by P_1, does not contain t and we use P_2 to denote the other one. Note that all three sites s, t, r are in P_2. According to Oh and Ahn [19] (see the proof of Lemma 3.6), α is in P_1 and $\overline{p_t\alpha}$ intersects $\overline{uv}$, i.e., p_t is in the pseudo-convex chain $\pi(t', r') \cup \pi(t', s')$; e.g., see Fig. 4.

Consider the bisector $B(s,t)$. Because α is equidistant from s and t, $\alpha \in B(s,t)$. As both s and t are in P_2, the middle point p_{st} of $\pi(s,t)$ is also in P_2. Note that $p_{st} \in B(s,t)$. As $\alpha \in P_1$ and $p_{st} \in P_2$, $B(s,t)$ must cross $\overline{uv}$; further, by Lemma 7, as $\overline{uv} \subseteq \pi(s,r)$, $B(s,t)$ intersects $\overline{uv}$ at a single point, denoted by q_{st} (e.g., see Fig. 4). Hence, q_{st} partitions $B(s,t)$ into two parts, one inside P_1 and the other in P_2. We use $B_1(s,t)$ to denote the part in P_1. Recall that $\beta(s,t)$ is on $B(s,t)$.

▶ **Lemma 9.** *$\beta(s,t)$ is on $B_1(s,t)$, and α is on $B_1(s,t)$ between $\beta(s,t)$ and q_{st} (e.g., see Fig. 4).*

Proof. First of all, since $\alpha \in P_1$, $\alpha \in B(s,t)$, and $B_1(s,t) = B(s,t) \cap P_1$, we obtain that $\alpha \in B_1(s,t)$. Notice that p_{st} is the point on $B(s,t)$ closest to t and if we move p from one end of $B(s,t)$ to the other end, $d(s,p)$ will first decrease until $p = p_{st}$ and then increase. By Lemma 6, if we move a point p along $B(s,t)$ from $\beta(s,t)$ to α, $d(t,p)$ decreases. Since $\alpha \in B_1(s,t)$, $\beta(s,t)$ must be on $B_1(s,t)$ and α is between $\beta(s,t)$ and q_{st}. ◀

Our algorithm relies on the following lemma to compute α (e.g., see Fig. 5).

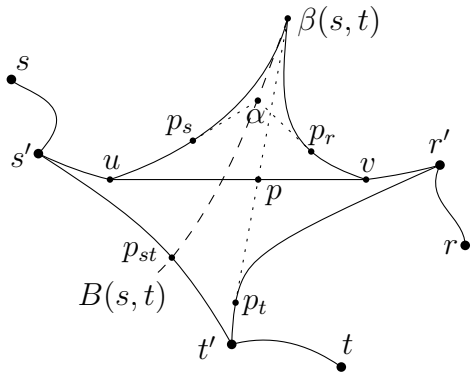


Figure 5 Illustrating the proof of Lemma 10.

▶ **Lemma 10.**

1. *α must be in the geodesic triangle $\triangle(s,r,\beta(s,t))$.*
2. *The apexes of $\triangle(s,r,\beta(s,t))$ are u, v, and $\beta(s,t)$.*
3. *p_s must be on the convex chain $\pi(u,\beta(s,t))$ and $\overline{\alpha p_s}$ is tangent to the chain.*
4. *p_r must be on the convex chain $\pi(v,\beta(s,t))$ and $\overline{\alpha p_r}$ is tangent to the chain.*
5. *p_t must be on the pseudo-convex chain $\pi(t_{uv},u) \cup \pi(t_{uv},v)$ and $\overline{\alpha p_t}$ is tangent to the chain, where t_{uv} is the junction vertex of $\pi(t,u)$ and $\pi(t,v)$.*
6. *$\overline{\alpha p_t}$ intersects $\overline{uv}$.*

Proof. To simplify notation, let $q = \beta(s,t)$. Since $q \in B(s,t)$, due to the general position assumption, the incident edges of q in $\pi(q,s)$ and $\pi(q,t)$ cannot be coincident [3]. Hence, q is the junction vertex of $\pi(q,s)$ and $\pi(q,t)$. As $t \in P_2$ and $q \in P_1$, $\pi(q,t)$ must cross $\overline{uv}$ at a point p; see Fig. 5. This implies that q is the junction vertex of $\pi(q,u)$ and $\pi(q,p)$. As $p \in \overline{uv}$, q must also be the junction vertex of $\pi(q,u)$ and $\pi(q,v)$. Since $u \in \pi(s,q)$ and $v \in \pi(q,r)$, q must be the junction vertex of $\pi(q,s)$ and $\pi(q,r)$. Hence, q is an apex of $\triangle(s,r,q)$.

Next we show that u and v are the other two apexes of $\triangle(s,r,q)$. By the definition of P_1, for any point $p' \in P_1$, $\pi(s,p')$ must contain u and $\pi(r,p')$ must contain v. Recall that $B(s,t)$ intersects $\overline{uv}$ at a single point q_{st}. Due to the general position assumption, q_{st} is in the interior of $\overline{uv}$. Hence, u and v are on different sides of $B(s,t)$. As $q \in B(s,t)$, $\pi(u,q)$ cannot contain v and $\pi(v,q)$ cannot contain u. Since $\pi(s,q)$ contains u, we obtain that $\pi(s,q) = \pi(s,u) \cup \pi(u,q)$ and thus $\pi(s,q)$ does not contain v. Similarly, since $\pi(r,q)$ contains v, we obtain that $\pi(r,q) = \pi(r,v) \cup \pi(v,q)$ and thus $\pi(r,q)$ does not contain u. Therefore, u is the junction vertex of $\pi(s,q)$ and $\pi(s,r)$; v is the junction vertex of $\pi(r,q)$ and $\pi(r,s)$. Thus, both u and v are apexes of $\triangle(s,r,q)$.

The above argument proves that the three apexes of $\triangle(s,r,q)$ are u, v, and q.

We proceed to show that $\alpha \in \triangle(s,r,q)$. Indeed, as α is on $B(s,t)$ between q and q_{st} by Lemma 9, α must be in the geodesic triangle $\triangle(s,t,q)$. As $\alpha \in P_1$, $\alpha \in P_1 \cap \triangle(s,t,q)$. Since $\pi(q,s) \cap P_1 = \pi(q,u)$ and $\pi(q,t) \cap P_1 = \pi(q,p)$, we have $P_1 \cap \triangle(s,t,q) = \triangle(u,p,q)$. Hence, $\alpha \in \triangle(u,p,q)$. Further, since $p \in \overline{uv}$, $\triangle(u,p,q) \subseteq \triangle(u,v,q)$. Thus, we obtain $\alpha \in \triangle(s,r,q)$, for $\triangle(s,r,q) = \triangle(u,v,q)$. This proves the first two lemma statements.

The lemma statement (6) was already proved in [19]. Finally we prove the lemma statements (3-5). Clearly, the anchor p_s must be on the two sides of $\triangle(s,r,q)$ incident to the apex u, i.e., $p_s \in \pi(u,q) \cup \overline{uv}$. Note that $\pi(u,\alpha)$ cannot contain v unless $\alpha = v$. However, $\alpha \neq v$ because α is not in $\triangle(s,t,r)$ (otherwise the first step of our algorithm, which is the algorithm for Lemma 5, would have already computed $\alpha(s,t,r)$). Hence, p_s must be on

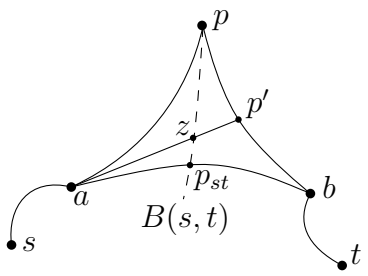


Figure 6 Illustrating the pseudo-triangle $\triangle(s,t,p)$.

$\pi(u,q)$, which is a convex chain. This also means that $\overline{\alpha p_s}$ is tangent to $\pi(u,q)$ at p_s. For the same reason, $\overline{\alpha p_r}$ is tangent to $\pi(v,q)$ at p_r. For p_t, because $\overline{\alpha p_t}$ intersects $\overline{uv}$, it is obviously true that $\overline{\alpha p_t}$ must be tangent to the pseudo-convex chain $\pi(t_{uv},u) \cup \pi(t_{uv},v)$ at p_t. ◀

In light of Lemma 10, $\overline{uv}$, it is actually tangent to $\pi(t',u) \cup \pi(t',v)$. Then, we can apply the tentative prune-and-search technique [15] on the three chains specified in the lemma in a similar way as before to compute α in $O(\log n)$ time.

We summarize our algorithm for this case. First, we compute the edge $\overline{uv}$, in $O(\log n)$ time using the GH data structure by Lemma 3. Second, we compute the junction vertex t_{uv} of $\pi(t,u)$ and $\pi(t,v)$ in $O(\log n)$ time by the GH data structure [11]. Third, we apply the tentative prune-and-search technique on the three chains as specified in Lemma 10, along with the binary search scheme in Lemma 3 on the chains, to compute α in $O(\log n)$ time.

Recall that the above algorithm is based on the assumption that α is a Voronoi vertex of $FVD(S)$. However, when we invoke the procedure during the geodesic sweeping algorithm we do not know whether the assumption is true. Therefore, as a final step, we add a validation procedure as follows. Suppose α is the point returned by the algorithm. First, we check whether $d(s,\alpha) = d(t,\alpha) = d(r,\alpha)$. If not, we return null. Otherwise, we further check whether $d(c^*,\alpha) \leq \min\{d(c^*,\beta(s,t)), d(c^*,\beta(t,r))\}$. This is because the Voronoi vertex α is only useful if it is inside the current sweeping circle C, whose geodesic distance to c^* is at most $\min\{d(c^*,\beta(s,t)), d(c^*,\beta(t,r))\}$ (because neither $\beta(s,t)$ nor $\beta(t,r)$ is in the interior of C). Hence, if $d(c^*,\alpha) \leq \min\{d(c^*,\beta(s,t)), d(c^*,\beta(t,r))\}$, then we return α; otherwise, we return null. This validation step takes $O(\log n)$ time by the GH data structure.

At least one of u and v is not a polygon vertex. The above discusses the case where both u and v are polygon vertices. In the following, we consider the other case where at least one of them is not a polygon vertex, i.e., $u = s$ or/and $v = r$ (because all vertices of $\pi(s,r)$ except s and r are polygon vertices). In fact, this case is missed from the algorithm of Oh and Ahn [19] (see the proof of Lemma 3.6 [19]). It turns out that Lemma 10 still holds for this case and thus we can apply exactly the same algorithm as above. By reducing this case to the previous case where u and v are polygon vertices. we can obtain the following lemma (whose proof is omitted).

▶ **Lemma 11.** *Lemma 10 still holds when $u = s$ or/and $v = r$.*

We finally prove the following technical lemma, which is needed in the proof of Lemma 11. The lemma, which establishes a very basic property of shortest paths in simple polygons, may be interesting in its own right. The proof is omitted.

▶ **Lemma 12.** *Let s and t be any two points in P such that $B(s,t)$ does not contain any vertex of P. Suppose p is a point in $B(s,t)$ and p' is a point in $\pi(t,p)$. Then, $\pi(s,p')$ intersects $B(s,t)$ at a single point z and $d(s,z) \geq d(z,p')$ (in particular, $d(s,z) > d(z,p')$ if $p' \neq t$); e.g., see Fig. 6.*

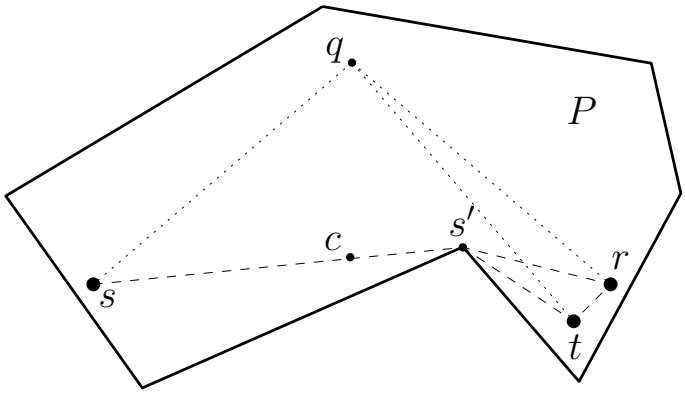


Figure 7 Illustrating an example where α exists when $c \in \pi(s, s') \setminus \{s'\}$. The apexes of the geodesic triangle $\triangle(s, r, t)$ are s', $r' = r$, and $t' = t$. $|\overline{s't}| < |\overline{s'r}|$. c is the middle point of $\pi(s, r) = \overline{ss'} \cup \overline{s'r}$. However, one can verify (e.g., by a ruler) that q is equidistant to s, r, and t, and thus $\alpha = q$ exists.

4.2.2 The case $c \in \pi(s, s') \setminus \{s'\}$

We now consider the case $c \in \pi(s, s') \setminus \{s'\}$. For this case, Oh and Ahn [19] (see the proof of Lemma 3.6 [19]) claimed that α does not exist. However, this is not correct; see Fig. 7 for a counterexample.

Let v be the vertex incident to s' in $\pi(s', r')$. To make the notation consistent with the previous subcase, we let $u = s'$. We have the following lemma (e.g., see Fig. 8), which is analogous to Lemma 10. The proof is omitted.

▶ **Lemma 13.**

1. *α must be in the geodesic triangle $\triangle(s, r, \beta(t, r))$.*
2. *The apexes of $\triangle(s, r, \beta(t, r))$ are s'', v, and $\beta(t, r)$, where s'' be the junction vertex of $\pi(s, r)$ and $\pi(s, \beta(t, r))$.*
3. *p_s must be on the pseudo-convex chain $\pi(s'', \beta(t, r)) \cup \pi(s'', v)$ and $\overline{\alpha p_s}$ is tangent to the chain.*
4. *p_r must be on the convex chain $\pi(v, \beta(t, r))$ and $\overline{\alpha p_r}$ is tangent to the chain.*
5. *p_t must be on pseudo-convex chain $\pi(t_{uv}, u) \cup \pi(t_{uv}, v)$ and $\overline{\alpha p_t}$ is tangent to the chain, where t_{uv} is the junction vertex of $\pi(t, u)$ and $\pi(t, v)$.*
6. *$\overline{\alpha p_t}$ must intersect $\overline{uv}$.*

Due to the preceding lemma, our algorithm works as follows. First, we compute the vertices s'', u, v, and t_{uv}, which can be done in $O(\log n)$ time by the GH data structure. Then we apply the tentative prune-and-search technique [15] on the three pseudo-convex chains specified in the lemma in a similar way as before to compute α in $O(\log n)$ time. Finally, we validate α in $O(\log n)$ time in a similar way as before. The overall time of the algorithm is $O(\log n)$. Lemma 1 is thus proved.

▶ Remark. As discussed above, there are two mistakes in the algorithm of Oh and Ahn [19] (Lemma 3.6): (1) In the subcase $c \in \pi(s', r')$, the case where not both u and v are polygon vertices is missed; (2) in the subcase $c \notin \pi(s', r')$, they erroneously claimed that α does not exist. Both mistakes can be corrected with our new results. Indeed, in both cases we have proved that $\overline{\alpha p_t}$ intersects $\overline{uv}$. With this critical property, their algorithm of Lemma 3.6 [19] (which was originally designed for the case where $c \in \pi(s', r')$ and both u and v are polygon vertices) can be applied to compute α in $O(\log^2 n)$ time. In this way, Lemma 3.6 of [19] is remedied and thus all other results of [19] that rely on Lemma 3.6 are not affected.

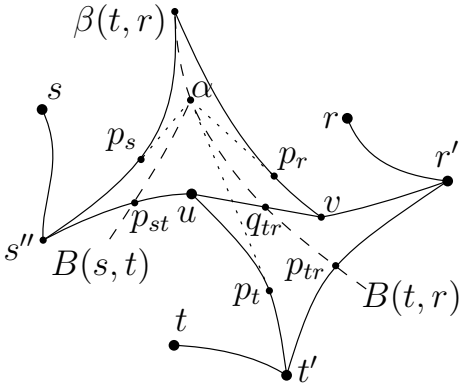


Figure 8 Illustrating Lemma 13.

References

1 H.-K. Ahn, L. Barba, P. Bose, J.-L. De Carufel, M. Korman, and E. Oh. A linear-time algorithm for the geodesic center of a simple polygon. *Discrete and Computational Geometry*, 56:836–859, 2016.

2 B. Aronov. On the geodesic Voronoi diagram of point sites in a simple polygon. *Algorithmica*, 4:109–140, 1989.

3 B. Aronov, S. Fortune, and G. Wilfong. The furthest-site geodesic Voronoi diagram. *Discrete and Computational Geometry*, 9:217–255, 1993.

4 T. Asano and G. Toussaint. Computing the geodesic center of a simple polygon. Technical Report SOCS-85.32, McGill University, Montreal, Canada, 1985.

5 S.W. Bae and K.Y. Chwa. The geodesic farthest-site Voronoi diagram in a polygonal domain with holes. In *Proceedings of the 25th ACM Symposium on Computational Geometry (SoCG)*, pages 198–207, 2009.

6 S.W. Bae, M. Korman, and Y. Okamoto. The geodesic diameter of polygonal domains. *Discrete and Computational Geometry*, 50:306–329, 2013.

7 S.W. Bae, M. Korman, and Y. Okamoto. Computing the geodesic centers of a polygonal domain. *Computational Geometry: Theory and Applications*, 77:3–9, 2019.

8 L. Barba. Optimal algorithm for geodesic farthest-point Voronoi diagrams. In *Proceedings of the 35th International Symposium on Computational Geometry (SoCG)*, pages 12:1–12:14, 2019.

9 B. Chazelle. A theorem on polygon cutting with applications. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 339–349, 1982.

10 B. Chazelle, H. Edelsbrunner, M. Grigni, L. Guibas, J. Hershberger, M. Sharir, and J. Snoeyink. Ray shooting in polygons using geodesic triangulations. *Algorithmica*, 12(1):54–68, 1994.

11 L.J. Guibas and J. Hershberger. Optimal shortest path queries in a simple polygon. *Journal of Computer and System Sciences*, 39(2):126–152, 1989.

12 J. Hershberger. A new data structure for shortest path queries in a simple polygon. *Information Processing Letters*, 38(5):231–235, 1991.

13 J. Hershberger and S. Suri. Matrix searching with the shortest-path metric. *SIAM Journal on Computing*, 26(6):1612–1634, 1997.

14 J. Hershberger and S. Suri. An optimal algorithm for Euclidean shortest paths in the plane. *SIAM Journal on Computing*, 28(6):2215–2256, 1999.

15 D. Kirkpatrick and J. Snoeyink. Tentative prune-and-search for computing fixed-points with applications to geometric computation. *Fundamenta Informaticae*, 22(4):353–370, 1995.

16 C.-H. Liu. A nearly optimal algorithm for the geodesic Voronoi diagram of points in a simple polygon. *Algorithmica*, 82:915–937, 2020.

17 J.S.B. Mitchell. Geometric shortest paths and network optimization, in *Handbook of Computational Geometry,* J.-R Sack and J. Urrutia (eds.), pages 633–702. Elsevier, Amsterdam, the Netherlands, 2000.

18 E. Oh. Optimal algorithm for geodesic nearest-point Voronoi diagrams in simple polygons. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 391–409, 2019.

19 E. Oh and H.-K. Ahn. Voronoi diagrams for a moderate-sized point-set in a simple polygon. *Discrete and Computational Geometry*, 63:418–454, 2020.

20 E. Oh, L. Barba, and H.-K. Ahn. The geodesic farthest-point Voronoi diagram in a simple polygon. *Algorithmica*, 82:1434–1473, 2020.

21 E. Papadopoulou and D.T. Lee. A new approach for the geodesic Voronoi diagram of points in a simple polygon and other restricted polygonal domains. *Algorithmica*, 20:319–352, 1998.

22 R. Pollack, M. Sharir, and G. Rote. Computing the geodesic center of a simple polygon. *Discrete and Computational Geometry*, 4(1):611–626, 1989.

23 F.P. Preparata and M.I. Shamos. *Computational Geometry.* Springer-Verlag, New York, 1985.

24 S. Suri. Computing geodesic furthest neighbors in simple polygons. *Journal of Computer and System Sciences*, 39:220–235, 1989.

25 G. Toussaint. Computing geodesic properties inside a simple polygon. Technical report, McGill University, Montreal, Canada, 1989.

26 H. Wang. On the geodesic centers of polygonal domains. *Journal of Computational Geometry*, 9:131–190, 2018.

A Parallel Batch-Dynamic Data Structure for the Closest Pair Problem

Yiqiu Wang
Massachusetts Institute of Technology, Cambridge, MA, USA

Shangdi Yu
Massachusetts Institute of Technology, Cambridge, MA, USA

Yan Gu
University of California, Riverside, CA, USA

Julian Shun
Massachusetts Institute of Technology, Cambridge, MA, USA

Abstract

We propose a theoretically-efficient and practical parallel batch-dynamic data structure for the closest pair problem. Our solution is based on a serial dynamic closest pair data structure by Golin et al., and supports batches of insertions and deletions in parallel. For a data set of size n, our data structure supports a batch of insertions or deletions of size m in $O(m(1+\log((n+m)/m)))$ expected work and $O(\log(n+m)\log^*(n+m))$ depth with high probability, and takes linear space. The key techniques for achieving these bounds are a new work-efficient parallel batch-dynamic binary heap, and careful management of the computation across sets of points to minimize work and depth.

We provide an optimized multicore implementation of our data structure using dynamic hash tables, parallel heaps, and dynamic k-d trees. Our experiments on a variety of synthetic and real-world data sets show that it achieves a parallel speedup of up to 38.57x (15.10x on average) on 48 cores with hyper-threading. In addition, we also implement and compare four parallel algorithms for static closest pair problem, for which we are not aware of any existing practical implementations. On 48 cores with hyper-threading, the static algorithms achieve up to 51.45x (29.42x on average) speedup, and Rabin's algorithm performs the best on average. Comparing our dynamic algorithm to the fastest static algorithm, we find that it is advantageous to use the dynamic algorithm for batch sizes of up to 20% of the data set. As far as we know, our work is the first to experimentally evaluate parallel closest pair algorithms, in both the static and the dynamic settings.

2012 ACM Subject Classification Theory of computation → Shared memory algorithms; Computing methodologies → Shared memory algorithms; Theory of computation → Computational geometry

Keywords and phrases Closest Pair, Parallel Algorithms, Dynamic Algorithms, Experimental Algorithms

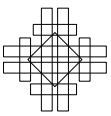
Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.60

Related Version *Full Version*: `https://arxiv.org/abs/2010.02379`

Supplementary Material *Software (Source Code)*: `https://github.com/wangyiqiu/closest-pair`

Funding This research was supported by DOE Early Career Award #DESC0018947, NSF CAREER Award #CCF-1845763, Google Faculty Research Award, DARPA SDH Award #HR0011-18-3-0007, and Applications Driving Architectures (ADA) Research Center, a JUMP Center co-sponsored by SRC and DARPA.

Acknowledgements We thank Yihan Sun for help on using the PAM library [45].



1 Introduction

The closest pair problem is a fundamental computational geometry problem with applications in robot motion planning [30, 3], computational biology [37], collision detection, hierarchical clustering, and greedy matching [18]. In many cases, the data involved can quickly change over time. In the case that a subset of the data gets updated, a dynamic algorithm can be much faster than a static algorithm that recomputes the result from scratch.

We consider a metric space (S, d) where S contains n points in $\mathbb{R}^k$, and d is the L_t-metric where $1 \leq t \leq \infty$. The **static closest pair** problem computes and returns the **closest pair distance** $\delta(S) = \min\{d(p,q) \mid p, q \in S, p \neq q\}$, and the closest pair (p, q). The **dynamic closest pair** problem computes the closest pair of S, and also maintains the closest pair upon insertions and deletions of points. A **parallel batch-dynamic** data structure processes batches of insertions and deletions of points of size m in parallel. In this paper, we propose a new parallel batch-dynamic data structure for closest pair on (S, d).

There is a rich literature on sequential dynamic closest pair algorithms [34, 46, 39, 42, 43, 40, 13, 27, 9, 24, 2]. More details on this related work is provided in [44] and the full version of this paper. However, as far as we know, none of the existing dynamic algorithms have been implemented and none of them are parallel. The main contribution of our paper is the design of a theoretically-efficient and practical parallel batch-dynamic data structure for dynamic closest pair, along with a comprehensive experimental study showing that it performs well in practice. Our solution is inspired by the sequential solution of Golin et al. [24], which takes $O(n)$ space to maintain $O(n)$ points and supports $O(\log n)$ time updates, and is the fastest existing sequential algorithm for the L_t-metric. Our parallel solution takes a batch update of size m and maintains the closest pair in $O(m(1 + \log((n+m)/m)))$ expected work and $O(\log(n+m)\log^*(n+m))$ depth (parallel time) with high probability (*whp*).[1] Compared to the sequential algorithm of Golin et al., our algorithm is work-efficient (i.e., matches the work of the sequential algorithm) for single updates, and has better depth for multiple updates since we process a batch of updates in parallel. Our data structure is based on efficiently maintaining a sparse partition of the points (a data structure used by Golin et al. [24]) in parallel. This requires carefully organizing the computation to minimize the work and depth, as well as using a new parallel batch-dynamic binary heap that we design. This is the first parallel batch-dynamic binary heap in the literature, and may be of independent interest.

We implement our data structure with optimizations to improve performance. In particular, we combine the multiple heaps needed in our theoretically-efficient algorithm into a single heap, which reduces overheads. We also implement a parallel batch-dynamic kd-tree to speed up neighborhood queries. We evaluate our algorithm on both real-world and synthetic data sets. On 48 cores with two-way hyper-threading, we achieve self-relative parallel speedups of up to 38.57x across various batch sizes. Our algorithm achieves throughputs of up to 1.35×10^7 and 1.06×10^7 updates per second for insertions and deletions, respectively.

In addition, we implement and evaluate four parallel algorithms for the static closest pair problem. There has been significant work on sequential [41, 36, 8, 20, 7, 25, 23, 28, 16, 14, 5] and parallel [4, 33, 32, 11, 10] static algorithms for the closest pair (more details can be found in [44] and the full version of the paper). As far as we know, none of the existing parallel algorithms have been evaluated and compared empirically. We implement a divide-and-conquer algorithm [11], a variant of Rabin's randomized algorithm [36], our parallelization of the sequential sieve algorithm [28], and a randomized incremental algorithm [10]. On 48

[1] Holds with probability at least $1 - 1/n^c$ for an input of size n and some constant $c > 0$.

cores with two-way hyper-threading, our algorithms achieve self-relative parallel speedups of up to 51.45x.[2] Our evaluation of the static algorithms shows that Rabin's algorithm is on average 7.63x faster than the rest of the static algorithms. Finally, we compare our parallel batch-dynamic algorithm with the static algorithms and find that it can be advantageous to use the batch-dynamic algorithm for batches containing up to 20% of the data set. Our source code is publicly available at `https://github.com/wangyiqiu/closest-pair`.

2 Review of the Sparse Partition Data Structure

We give an overview of the sequential dynamic closest pair data structure proposed by Golin et al. [24], which is based on the serial static closest pair algorithm by Khuller and Matias [28]. More details are presented in the full version of the paper. Our new parallel algorithm also uses this data structure, which is referred to as the *sparse partition* of an input set.

2.1 Sparse Partition

For a set S with n points, a **sparse partition** [24] is defined as a sequence of 5-tuples $(S_i, S'_i, p_i, q_i, d_i)$ with size L $(1 \leq i \leq L)$, such that **(1)** $S_1 = S$; **(2)** $S'_i \subseteq S_i \subseteq S$; **(3)** If $|S_i| > 1$, p_i is drawn uniformly at random from S_i, and we compute the distance $d_i = d(p_i, q_i)$, to p_i's closest point q_i in S_i; **(4)** For all $x \in S_i$: **a)** if the closest point of x in S_i (denoted as $d(x, S_i)$) is larger than $d_i/3$, then $x \in S'_i$; **b)** if $d(x, S_i) \leq d_i/6k$ then $x \notin S'_i$; and **c)** if $x \in S_{i+1}$, there is a point $y \in S_i$ such that $d(x, y) \leq d_i/3$ and $y \in S_{i+1}$; **(5)** $S_{i+1} = S_i \setminus S'_i$.

The sparse partition is constructed using these rules until $S_{L+1} = \emptyset$. It contains $O(\log n)$ levels in expectation, as $|S_i|$ decreases geometrically. The expected sum of all $|S_i|$ is linear [24]. We call p_i the **pivot** for partition i. At a high level, S'_i contains points that are far enough from each other, and the threshold d_i that defines whether points are "far enough" decreases for increasing i. In particular, for any $1 \leq i < L$, $d_{i+1} \leq d_i/3$ as shown in Golin et al. [24]. Hence, the closest pair will likely show up in deeper levels that do not contain many points. Based on the definition, each S'_i is non-empty, and $\{S'_1, \ldots, S'_L\}$ is a partition of S.

2.2 A Grid-Based Implementation of Sparse Partition

We now describe Golin et al.'s implementation of the sparse partition. There are L levels of the sparse partition, and we refer to each as **level** i for $1 \leq i \leq L$. We maintain each level using a grid data structure, similar to many closest pair algorithms (e.g., [36, 23, 28, 24]).

To represent S_i, we place the points into a grid G_i with equally-sized axis-aligned grid boxes with side length $d_i/6k$, where k is the dimension, and d_i is the closest pair distance of the randomly chosen pivot p_i. This can be done using hashing. Denote the **neighborhood** of a point p in G_i relative to S by $N_i(p, S)$, which refers to the set of points in $S \setminus \{p\}$ contained in the collection of $3^k - 1$ boxes bordering the box containing p, as well as p's box. We say that point p is **sparse** in G_i relative to S if $N_i(p, S) = \emptyset$. We use this notion of sparsity to compute $S'_i = \{p \in S_i : p \text{ is sparse in } G_i \text{ relative to } S_i\}$, which satisfies definition (4) of the sparse partition. The points in S'_i are stored in a separate grid.

An example of the grid structure in two dimensions is shown in Figure 1. We illustrate the grid G_i for the S_i of each level, as well as the pivot p_i and its closest neighbor q_i. The grid size is set to $d_i/6k = d(p_i, q_i)/12$. The sparse points, represented by the hollow blue

[2] With hyper-threading, the parallel speedup can be more than the total core count.

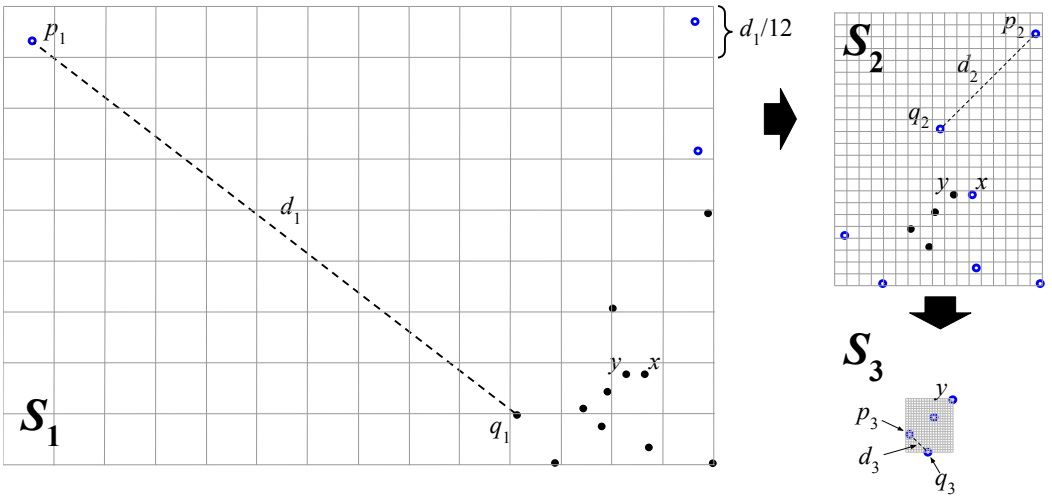


Figure 1 This figure contains an example of 14 points in $\mathbb{R}^2$, for which a grid-based sparse partition $(S_i, S'_i, p_i, q_i, d_i)$ for $1 \leq i \leq 3$ is constructed. On each level, we use a dotted line to indicate d_i, the Euclidean distance between the pivot p_i and its closest neighbor q_i, and we set the grid size to be $d_i/6k = d_i/12$. We denote non-sparse points as solid black circles and sparse points as hollow blue circles. The S'_i sets are represented implicitly by the set of hollow blue circles in each S_i. We denote the true closest pair by letters x and y.

circles, have empty neighborhoods, and do not have another point within a distance of $d_i/3$. The solid black circles, representing the non-sparse points, are copied to the grid G_{i+1} for S_{i+1}. In S_3, all points are sparse.

To construct a sparse partition, the sequential algorithm proceeds in rounds. On round i, the i'th level is constructed. We start with $i = 1$ where $S_1 = S$, and iteratively determine the side length of grid G_i based on a random pivot, and place S_i into G_i. Then, we compute S'_i based on the definition of sparsity above, and set $S_{i+1} = S_i \setminus S'_i$. The algorithm proceeds until $S_i = S'_i$ (i.e., $S_{i+1} = \emptyset$). The expected work for construction is $O(n)$ since $|S_i|$ decreases geometrically [24]. A single insertion of point q starts from S_1, and proceeds level by level. When q is non-sparse in S_i, it will be added to S_{i+1}, and can promote points from S'_i to S'_{i+1} if q falls within their neighborhood. The insertion of q will stop if it becomes sparse at some level, at which point the insertion algorithm finishes. A deletion works in the opposite direction, starting from the last level where the deleted point exists, and working its way back to level 1. Each insertion or deletion takes $O(\log n)$ expected work.

2.3 Obtaining the Closest Pair

As observed by both Khuller and Matias [28] and Golin et al. [24], although the grid data structure rejects far pairs, and becomes more fine-grained with a larger i, the grid at the last (L'th) level does not necessarily contain the closest pair. For example, as illustrated in Figure 1, S_3 for the last level does not contain the closest pair (x, y), as x is sparse on level 2 and not included in S_3. Therefore, we need to check more than just the last level.

The **restricted distance** $d^*_i(p)$ [24] is the closest pair distance from point p to any point in $\bigcup_{0 \leq j \leq k} S'_{i-j}$, and defined as $d^*_i(p) := \min\{d_i, d(p, S'_{i-k} \cup S'_{i-k+1} \cup \ldots \cup S'_i)\}$, where $p \in S'_i$. Golin et al. show that $\delta(S) = \min_{L-k \leq i \leq L} \min_{p \in S'_i} d^*_i(p)$, meaning that the closest pair can be found by taking the minimum among the restricted distance pairs for all points in the last $k+1$ levels of S'_i. The sequential algorithm [24] computes the restricted distance for each

Algorithm 1 Construction.

Input : Point set S.
Output : A sparse partition and its associated heaps.

Algorithm MAIN()
  BUILD(S, 1); /* Initially, $S_i := S$. */
Procedure BUILD(S_i, i)
  Choose a random point $p_i \in S_i$. Calculate $d_i := d(p_i, S_i)$, set the grid side length to $d_i/6k$, and store p_i's nearest neighbor as q_i.
  Create a parallel dictionary S_i^{dict} to store points in S_i keyed by box ID. In parallel, compute the box ID of each point in S_i based on the grid size, and store the point in the box keyed by the box ID in S_i^{dict}.
  Create a parallel dictionary $S_i'^{\mathsf{dict}}$ to store points in S_i' keyed by box ID. In parallel, determine if each point x in S_i is sparse by checking $N_i(q, S_i)$ (using S_i^{dict}). Store the sparse points in S_i' and $S_i'^{\mathsf{dict}}$, and the remaining points in a new point set represented by an array S_{i+1}.
  In parallel for each point $x \in S_i'$, compute $d_i^*(x)$ by checking its neighborhoods (using $S_j'^{\mathsf{dict}}$) in S_j' where $i - k \le j \le i$.
  fork Create a heap for $\{d_i^*(x) : x \in S_i'\}$.
    BUILD(S_{i+1}, $i+1$) if S_{i+1} is not empty.
  join

point in S_i', and stores it in a min-heap H_i for level i. To obtain the closest pair, we read the minimum of H_i for $L-k \le i \le L$, and then take the overall minimum. This takes $O(1)$ work. In the full version of the paper, we provide more background on the restricted distance.

3 Computational Model

We use the classic **work-depth model** for analyzing parallel algorithms [15, 26]. The **work** W of an algorithm is the number of instructions in the computation, and the **depth** D is the length of the longest sequential dependence. Using Brent's scheduling theorem [12], we can execute a parallel computation in $W/p + D$ running time using p processors. A parallel algorithm is **work-efficient** if its work asymptotically matches the work of the best sequential algorithm for the same problem. We assume that arbitrary concurrent writes are supported in $O(1)$ work and depth. Our pseudocode uses the **fork** and **join** keywords for fork-join parallelism [15]. A **fork** creates a task that can be executed in parallel with the current task, and a **join** waits for all tasks forked by the current task to finish. In the full version of the paper, we present details on the parallel primitives that we use.

For our batch-dynamic data structure, we assume that the updates are independent of the random choices made in our data structure.

4 Parallel Batch-Dynamic Data Structure

In this section, we introduce our parallel batch-dynamic algorithm, including the construction of the sparse partition (defined in Section 2) and how to handle batch updates.

As shown in Algorithm 1, we start with an initial point set S, on which we construct a grid structure recursively level by level, until all points become sparse. Starting with $S_i = S$, the algorithm works on point set S_i for level i. We first pick a pivot point $p_i \in S_i$

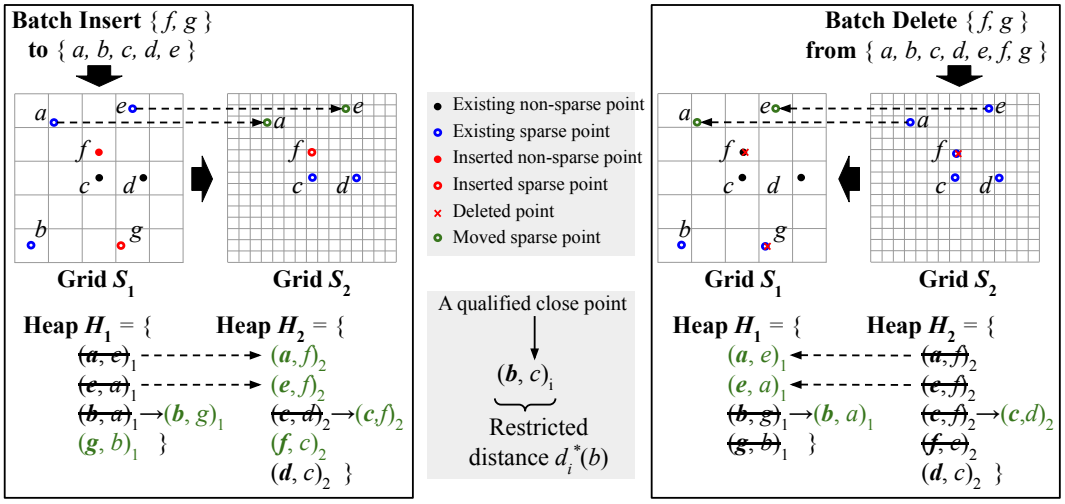


Figure 2 The figure illustrates the interaction between our parallel batch-dynamic insertion (left) and deletion (right) algorithms with the data structure. For ease of illustration, we do not show all the points in the data set. We show the data structure with two levels, and explicitly show S_i and H_i for each level. The grid structure in the upper half of the figures determines the sparsity of points. We represent different types of points as defined in the middle legend. In the lower half of the figures, we show the heaps with the restricted distances that they store. We show the pair defining the restricted distance of a sparse point x on level i as $(\boldsymbol{x}, y)$ if another point y is the closest sparse point to x in levels $i - j$ where $0 \leq j \leq 2$. For both insertion and deletion, we annotate the direction of the update between grids using large bold arrows (i.e., insertion starts with S_1 and deletion starts with S_2). We indicate the movement of points and heap entries using dotted arrows.

and obtain its closest pair by computing distances to all other points in parallel, followed by computing the minimum to determine the side length of the grid boxes, which we use a parallel dictionary [22] to store (Lines 4–5). We check the sparsity of each point x in parallel by looking up neighboring boxes using the dictionary, and store the sparse points S'_i in a new parallel dictionary and the remaining points in a new point set array S_{i+1} (Line 6). Then, we compute the restricted distances of all points in S'_i in parallel (Line 7), and spawn a thread to asynchronously construct the heap H_i to store the restricted distances (Line 8). We recursively call the construction procedure on S_{i+1} to construct the next level until all points in a level are sparse (Line 9). In the full version of the paper, we include more details about the algorithm, prove a high probability bound on the number of levels, and explain how to achieve optimal linear work and space. We summarize our bounds in the following theorem.

▶ **Theorem 1.** *We can construct a data structure that maintains the closest pair containing n points in $O(n)$ expected work, $O(\log n \log^* n)$ depth whp, and $O(n)$ expected space.*

Next, we present our parallel algorithm that processes a batch Q of m insertions or deletions. For $m \geq n$, we can simply rebuild the data structure on all of the points using Theorem 1 to obtain the desired bounds. We now describe the case for $m < n$. For batch updates, there are two main tasks: updating the grid and updating the heap. We first describe updating the grid. We let Q_i be the subset of points in Q that are inserted at level i, and $down_i$ be the set of points that move from level $i - 1$ to level i due to the insertion of Q_i. We start with a simple example of an insertion in Figure 2 (left), which originally contains five points $\{a, b, c, d, e\}$. For simplicity, we assume that the pivot remains unchanged

Algorithm 2 Batch Insert.

Input : $(S_i, S'_i, p_i, q_i, d_i)$ and H_i for $1 \le i \le L$; a batch Q to be inserted.

1 **Algorithm** MAIN()
2 | INSERT(Q, $\emptyset$, 1);
3 **Procedure** INSERT(Q_i, $down_i$, i)
4 | $(Q_{i+1}, down_{i+1})$:= GRIDINSERT(Q_i, $down_i$, i);
5 | HEAPUPDATE(i);
6 | **if** $(Q_{i+1} \cup down_{i+1}) \neq \emptyset$ **then** INSERT(Q_{i+1}, $down_{i+1}$, $i+1$) ;
7 **Procedure** GRIDINSERT(Q_i, $down_i$, i)
8 | Determine if p_i, q_i, and d_i should change when inserting Q_i and $down_i$, which happens with probability $(|Q_i| + |down_i|)/(|Q_i| + |down_i| + |S_i|)$, or if a new point is closer to p_i than the previously closest point q_i.
9 | If p_i, q_i, or d_i change on Line 8, or if $i > L$, call BUILD($Q_i \cup down_i \cup S_i, i$) to rebuild subsequent levels, and terminate the batch insertion.
10 | Insert each point in $down_i$ and Q_i into the dictionary of S_i in parallel.
11 | For each point x in Q_i in parallel, check if it is sparse in S_i. If so, insert x into the dictionary of S'_i, and otherwise, insert x into Q_{i+1}.
12 | For each point x in $down_i$ in parallel, check if it is sparse in S_i. If so, insert x into the dictionary of S'_i, and otherwise, insert x into $down_{i+1}$.
13 | In parallel, for each point x in Q_i, and for each point r in the neighborhood $N(x, S'_i)$, delete r from S'_i, and insert r into $down_{i+1}$.
14 | **return** $(Q_{i+1}, down_{i+1})$;

and also omits S'_i. $Q_1 = \{f, g\}$ is the set of points inserted into the grid at level 1. We first update S_1 to include f and g, and then update S_2 to include $Q_2 = \{f\}$ but not g, since g is already sparse in S_1. In the example, the insertion of Q_1 triggers further point movements of $\{a, e\}$ from level 1 to level 2, as the sparse points a and e in S_1 become non-sparse due to the insertion of f.

We explain the insertion algorithm in detail, and defer most details of the deletion algorithm to the full version of the paper. As shown in Algorithm 2, the update proceeds recursively level by level (Lines 3–6). Each call to the procedure INSERT(Q_i, $down_i$, i) updates $(S_i, S'_i, p_i, q_i, d_i)$ and H_i. Initially, $Q_1 = Q$ and $down_1$ is empty, as shown on Line 2. For each level i, we update the pivot and rebuild the level with probability $(|Q_i| + |down_i|)/(|Q_i| + |down_i| + |S_i|)$ to ensure that the pivot is still selected uniformly at random among the points in S_i, and we also update the pivot if a new point is closer to p_i than the previous closest point q_i (Lines 8–9). Otherwise, we insert the points in both $down_i$ and Q_i into the dictionary representing S_i. We then check if the points that we inserted are sparse, and insert the sparse ones into the dictionary representing S'_i. The points that are not sparse will be added to sets $down_{i+1}$ and Q_{i+1} and passed on to the next level (Lines 10–12). We then determine additional elements of $down_{i+1}$ by including the points in the neighborhood of Q_i in S'_i (Line 13). In general, $down_{i+1}$ is computed by $down_{i+1} = \{x \mid x \in N_i(q, S'_i \cup down_i)$ for some $q \in Q_i\}$. If Q_{i+1} and $down_{i+1}$ are empty, nothing further needs to be done for subsequent levels, and the tuples $(S_l, S'_l, p_l, q_l, d_l)$ for $i < l \le L$ remain unchanged.

We now argue that the algorithm is correct. Consider a round i that inserts a non-empty $Q_i \cup down_i$. After the insertion, the pivot is still chosen uniformly at random, since on Line 8, we choose p_i such that each point in $S_i \cup Q_i \cup down_i$ has the same probability of being chosen. All sparse points in Q_i and $down_i$ inserted into S_i are included in S'_i (Lines 11–12). Line 13

additionally ensures that all points that were originally sparse in S'_i, but are no longer sparse after the insertion, are removed from S'_i. Given that the non-sparse points in the original S_i will not become sparse due to the batch insertion, S'_i must contain exactly all of the sparse points of the updated S_i.

In our algorithm, each point can be moved across multiple levels as a result of a batch insertion. In the full version of the paper, we prove the following lemmas, which are key to maintaining work-efficiency.

▶ **Lemma 2.** $|\bigcup_{1\le i\le L} down_i| \le m \cdot 3^k = O(m)$

▶ **Lemma 3.** $\sum_{1\le i\le L} E[|Q_i|] = O(m)$

Deletions work similarly in the reverse direction as shown in Figure 2 (right), and we provide more details in the full version of the paper. We obtain Lemmas 4 and 5.

▶ **Lemma 4.** *We can maintain a sparse partition for a batch of m insertions in $O(m)$ amortized work in expectation and $O(\log(n+m)\log^*(n+m))$ depth whp.*

Proof. The expected cost of rebuilding on Line 9 summed across all rounds is proportional to the batch size. First, we re-select the pivot and rebuild with probability $(|Q_i|+|down_i|)/(|S_i|+|Q_i|+|down_i|)$. When the pivot p_i is unchanged, it may update its closest point to q_i^* from $Q_i \cup down_i$. It is easy to show that q_i^* can be the nearest neighbor of at most 3^k-1 points in S_i. Hence, considering all candidates $Q_i \cup down_i$, it follows that they can be the nearest neighbors to $O(3^k \cdot (|Q_i|+|down_i|))$ points in S_i. Therefore, the pivot distance changes with probability at most $3^k \cdot (|Q_i|+|down_i|)/|S_i|$, in which case we rebuild the sparse partition. The expected work of rebuilding at level i is $O((|S_i|+|Q_i|+|down_i|) \cdot ((|Q_i|+|down_i|)/(|S_i|+|Q_i|+|down_i|) + 3^k \cdot (|Q_i|+|down_i|)/|S_i|)) = O(m)$. As we terminate the insertion algorithm when a rebuild occurs, the rebuild can occur at most once for each batch, which contributes $O(m)$ in expectation to the work and $O(\log(n+m)\log^*(n+m))$ *whp* to the depth by Theorem 1.

For the rest of the algorithm, in terms of work, Line 10–12 does work proportional to $O(\sum_i(|Q_i|+|down_i|)) = O(m)$ across all the levels due to Lemmas 2 and 3. On Line 13, the number of points in the neighborhood $N_i(x, S'_i)$ of each x is upper bounded by 3^k since the points in S'_i are sparse, therefore it takes $O(3^k \cdot m) = O(m)$ expected work. Note that the work is amortized due to resizing the parallel dictionary when necessary. In terms of depth, looking up and inserting points takes $O(\log^*(n+m))$ depth using the parallel dictionary. Therefore, all operations in Lines 10–13 takes $O(\log^*(n+m))$ depth, and across all $O(\log(n+m))$ *whp* rounds, the total depth is $O(\log(n+m)\log^*(n+m))$ *whp*. ◀

▶ **Lemma 5.** *We can maintain a sparse partition for a batch of m deletions in $O(m)$ amortized work in expectation and $O(\log(n+m)\log^*(n+m))$ depth whp.*

Now we describe the parallel updates of min-heaps H_i associated with each level i of the sparse partition. Recall that H_i contains the restricted distances $d_i^*(q)$ for $q \in S'_i$. By definition, $d_i^*(q)$ is the closest distance of q to another point in S'_{i-l} where $0 \le l \le k$ (k is the dimensionality). Therefore, following an update on S'_i, we need to update the $d_i^*(q)$'s in H_{i+l} for $0 \le l \le k$, and $q \in S'_{i+l}$. We use same example in Figure 2 (left), where we denote the restricted distance of point x as $(\boldsymbol{x}, y)_i = d_i^*(x) = d(x,y)$, where $y \in \bigcup_{0\le j\le k} S'_{i-j}$ is another point that defines x's closest distance. As shown in Figure 2 (left), due to the insertion of the sparse point g to S_1, entry $(\boldsymbol{g}, b)_1$ is added to H_1. Some entries in H_1 are moved due to

the point movements, e.g., $(\boldsymbol{a}, e)_1$ from H_1 is moved and updated to $(\boldsymbol{a}, f)_2$ in H_2 because a has moved from S_1 to S_2, and f is now closer. Some entries are updated, e.g., $(\boldsymbol{c}, d)_2$ is updated to $(\boldsymbol{c}, f)_2$ in H_2 since the new point f is closer to c than d.

Our algorithm uses a new parallel batch-dynamic binary heap that we introduce in Section 5. For a batch-parallel binary heap of size n and a batch update (a mix of inserts, deletes, and increase/decrease-keys) of size m, updating the heap takes $O(m \log((n+m)/m))$ work and $O(\log(n+m))$ depth, and find-min takes $O(1)$ work. A simple implementation of our algorithm would execute all min-heap updates for an updated S'_i before processing the updates for S'_j where $j > i$ for insertions and $j < i$ for deletions. This level-by-level dependence leads to $O(\log^2(n+m))$ depth overall for the heap updates. In the full version of the paper, we present an improved algorithm that breaks the level dependencies and enables updates to be performed to a min-heap as early as possible, leading to $O(\log(n+m))$ depth overall for the heap updates. Together with Lemma 4 and Lemma 5, we obtain the following theorem, whose analysis we present in the full version of the paper.

▶ **Theorem 6.** *Updating our data structure for a batch of m insertions/deletions takes amortized $O(m(1+\log((n+m)/m)))$ expected work and $O(\log(n+m)\log^*(n+m))$ depth whp.*

Obtaining the closest pair from our data structure takes $O(1)$ work and depth. We simply call find-min on H_i for $L-k \leq i \leq L$, and then take the overall minimum.

5 Parallel Batch-Dynamic Binary Heap

One of the key components in parallelizing our closest pair algorithm is a parallel binary heap that supports batch updates (inserts and deletes) and find-min efficiently. This heap allows us to perform the parallel construction in linear work and perform updates with low depth. Our data structure may be of independent interest, since to the best of our knowledge, the only existing work on parallelizing a binary heap is on individual inserts or deletes [35].

A binary heap is a complete binary tree, where each node contains a key that is smaller than or equal to the keys of its children. Sequentially, the construction of a binary heap takes linear work, and each insert and delete takes $O(\log n)$ work [15]. The heap is represented as an array, and uses relative positions within the array to represent child-parent relationships. Sequentially, each insertion adds a new node at the end of the heap and runs Up-Heap to propagate the node up to the correct position in the heap. A deletion first swaps the node to delete with the node to the end of the heap, reduces the heap size by one, and then runs Up-Heap followed by Down-Heap (to propagate a node down to its correct position) for the node swapped to the middle of the heap.

Central to our parallel batch-dynamic binary heap is a new parallel Heapify algorithm, that takes m updates from a valid heap of n elements, and returns another valid heap. It runs in two phases: the first phase works on increase-key updates, and the second phase on decrease-key updates. In both phases, we first use parallel integer sorting [38, 47] to categorize all updates based on the level where the update belongs. Simply running the Up-Heap and Down-Heap calls for the different updates in parallel does not achieve work-efficiency and low depth, and also leads to potential data races. Therefore, we *pipeline* each level of the Up-Heap and Down-Heap procedure. Specifically, in the first phase, once the first swap for the Down-Heap in level i is finished, we can immediately start the Down-Heap on level $i-1$, instead of waiting for the Down-Heap in level i to completely finish (the root is at level 0, and level numbers increase going down). The swaps in the Down-Heap calls from

level $i-1$ will never catch up with the swaps from level i. Pipelining the second phase with Up-Heap is more complicated. Our parallel Up-Heap is run in a level-synchronous manner from the top level down to the bottom level. For each node on each level, both of its children may want to swap with the parent for having a larger value. In the parallel algorithm, we only make the child with the smaller value swap with the parent and continue its update to the upper levels, while the update for the other child terminates. We prove in the full paper that our parallel Heapify algorithm takes $O(m(1+\log(n/m)))$ work and $O(\log n)$ depth.

We now explain how to perform batch insertions and deletions. A batch of m insertions to a binary heap of size n can be implemented using decrease-keys. We first add the m elements to end of the heap with keys of ∞. Then, we decrease the keys of these m elements to their true values and run the parallel Heapify algorithm. A batch of m deletions can be processed similarly, but the deletions will generate "holes" in the tree structure, and so we need an additional step to fill these holes first. We pack the last m elements in the heap based on whether they are deleted. Then, we use them to fill the rest of the empty slots by deletions, and run the parallel Heapify algorithm. Hence, batch insertions and deletions take $O(m(1+\log((n+m)/m)))$ work and $O(\log(n+m))$ depth.

We provide more details about our data structure and prove the following theorem in the full version of the paper.

▶ **Theorem 7.** *For a batch-parallel binary heap of size n and a batch update (a mix of inserts, deletes, and increase/decrease-keys) of size m, updating the heap takes $O(m(1+\log((n+m)/m)))$ work and $O(\log(n+m))$ depth, and find-min takes $O(1)$ work.*

6 Implementations

Simplified Data Structure. While the sparse partition maintains $(S_i, S'_i, p_i, q_i, d_i)$ and H_i for each level $1 \le i \le L$, we found that implementing S'_i and its associated heap H_i on every level was inefficient in practice. We found it more efficient to only maintain (S_i, p_i, q_i, d_i) for $1 \le i \le L$, and one heap H^* that stores the closest neighbor distances for all q in S_j, where $j = L - \lceil \log_3 2\sqrt{k} \rceil$. When L changes due to insertion or deletion, we recompute j and rebuild H^* if necessary. We prove in the full version of the paper that H^* contains the closest pair. Our implementation uses the parallel heap from [45]. Additionally, we compute S'_i from S_i on the fly when needed.

Neighborhood Search. Some of the work bounds are exponential in the dimensionality k, e.g., a grid's box neighborhood is of size 3^k. For $k \ge 5$, the straightforward implementation is inefficient due to a large constant overhead in the work. Hence, we implement a parallel batch-dynamic kd-tree for $k \ge 5$. This is because performing a range query on the tree works better in practice, as it only needs to traverse the non-empty boxes in the neighborhood instead of all boxes. Our dynamic kd-tree is a standard spatial median kd-tree [6], augmented with the capability for parallel batch updates. Each internal node maintains metadata on the points in its subtree, which are partitioned by a spatial median along the widest dimension. The points are only stored at leaf nodes. We flatten a subtree to a single leaf node when it contains at most 16 points.

The tree supports batch insertion by first adding the batch to the root, and then traversing down multiple branches of the tree in parallel. At each internal node, we partition the inserted batch by the spatial median stored at the node, and modify its metadata, such as the point count and the coordinates of its bounding box. At each leaf node, we directly modify the

metadata and store the points. The tree supports batch deletions by modifying the metadata, and marking the deleted points at the leaves as invalid. We manage the memory periodically to free up the invalid entries.

Static Algorithms. In addition to our batch-dynamic closest pair algorithm, we implement several sequential and parallel algorithms for the static closest pair problem. As far as we know, this paper presents the first experimental study of parallel algorithms for static closest pair. We implement a parallel divide-and-conquer algorithm by Blelloch and Maggs [11], a simplified and parallel version of Rabin's algorithm [36] that we designed, a parallel version of Khuller and Matias's [28] sieve algorithm that we designed, and a parallel randomized incremental algorithm by Blelloch et al. [10]. We explain more details about these static algorithms and their implementations in the full paper.

7 Experiments

Algorithms Evaluated. We evaluate our parallel batch-dynamic algorithm by benchmarking its performance on batch insertions (**dynamic-insert**) and batch deletions (**dynamic-delete**). We also evaluate the four static implementations described in Section 6, which we refer to as **divide-conquer**, **rabin**, **sieve**, and **incremental**. In addition, we implement and evaluate sequential versions of all of our algorithms that do not have the overheads of parallelism. Our implementations use the Euclidean metric (L_2-metric).

Data Sets. We use the synthetic seed spreader (SS) data sets produced by the generator in [21]. It produces points generated by a random walk in a local neighborhood, but jumping to a random location with some probability. **SS-varden** refers to the data sets with variable-density clusters. We also use a synthetic data set called **Uniform**, in which points are distributed uniformly at random inside a bounding hyper-cube with side length $\sqrt{n}$, where n is the total number of points. The points have double-precision floating-point values. We generated the synthetic data sets with 10 million points for dimensions $k = 2, 3, 5, 7$. We name the data sets in the format of **Dimension-Name-Size**. We also use the following real-world data sets: **7D-Household-2M** [17] is a 7-dimensional data set containing household sensor data with $2,049,280$ points excluding the date-time information; **16D-Chem-4M** [19, 1] is a 16-dimensional data set with $4,208,261$ points containing chemical sensor data; and **3D-Cosmo-298M** [29] is a 3-dimensional astronomy data set with $298,246,465$ points.

Testing Environment. Our experiments are run on an `r5.24xlarge` instance on Amazon EC2. The machine has 2 × Intel Xeon Platinum 8259CL CPU (2.50 GHz) CPUs for a total of 48 cores with two-way hyper-threading, and 768 GB of RAM. By default, we use all cores with hyper-threading. We use the `g++` compiler (version 7.5) with the `-O3` flag, and use Cilk Plus for parallelism [31]. We use the **-48h** and **-1t** suffixes in our algorithm names to denote the 48-core with hyper-threading and single-threaded times, respectively. We allocate a maximum of 2 hours for each test, and do not report times for tests that exceed this limit.

Influence of Batch Size on Throughput. In this experiment, we evaluate our batch-dynamic algorithm by measuring their throughput as a function of the batch size. For insertions, we insert batches of the same size until the entire data set is inserted. For deletions, we start with the entire data set and delete batches of the same size until the entire data set is deleted. We compute throughput by the number of points processed per second. We vary the batch

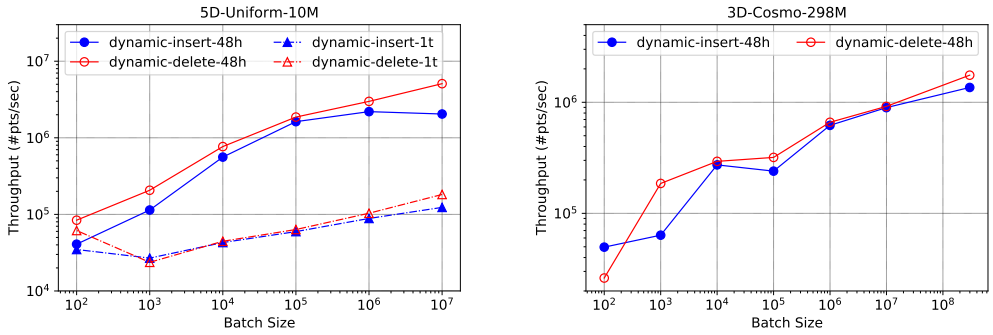


Figure 3 Plots of throughput vs. batch size in log-log scale for our parallel batch-dynamic algorithm on 5D-Uniform-10M and 3D-Cosmo-298M. The algorithm on 48-cores with hyper-threading and 1 thread has a suffix of "48h" and "1t", respectively. For 3D-Cosmo-298M, we omit the 1-thread times as the experiments exceeded our time limit.

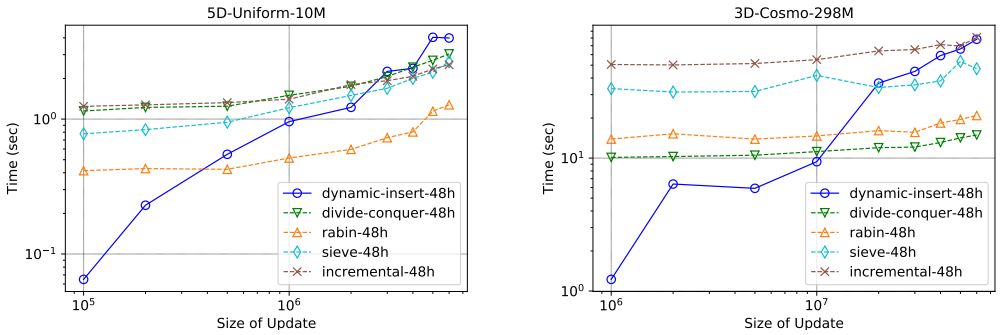


Figure 4 Plots of running time (in seconds) vs. insertion batch size for the dynamic and static methods using 48 cores with hyper-threading on 5D-Uniform-10M and 3D-Cosmo-298M. The plots are in log-log scale.

size from 100 points to the size of the entire data set. Our parallel batch-dynamic algorithm achieves a throughput of up to 1.35×10^7 points per second for insertion, and up to 1.06×10^7 for deletion, under the largest batch size. On average, it achieves 1.75×10^6 for insertion and 1.94×10^6 for deletion across all batch sizes. We show plots of throughput vs. batch size for 5D-Uniform-10M and 3D-Cosmo-298M in Figure 3. We see that the throughput increases with larger batch sizes because of a lower relative overhead of traversing the sparse partition data structure, and the availability of more parallelism.

Efficiency of Batch Insertions. We evaluate the performance of dynamic batch insertion vs. using a static algorithm to recompute the closest pair. Specifically, we simulate a scenario where given the data structure storing the closest pair among c data points, we perform an insertion of b additional points. We compare the time taken by the dynamic algorithm to process one batch insertion of size b, vs. that of a static algorithm for recomputing the closest pair for all $c + b$ points. We set c to contain 40% of the data set and vary b. Figure 4 shows the running time as a function of b for 5D-Uniform-10M and 3D-Cosmo-298M. For 5D-Uniform-10M, we see that our batch-dynamic algorithm outperforms the fastest among the static algorithms when the insertion batch size is smaller than 500,000. For 3D-Cosmo-298M, we see that the dynamic method outperforms the fastest static algorithm when the insertion

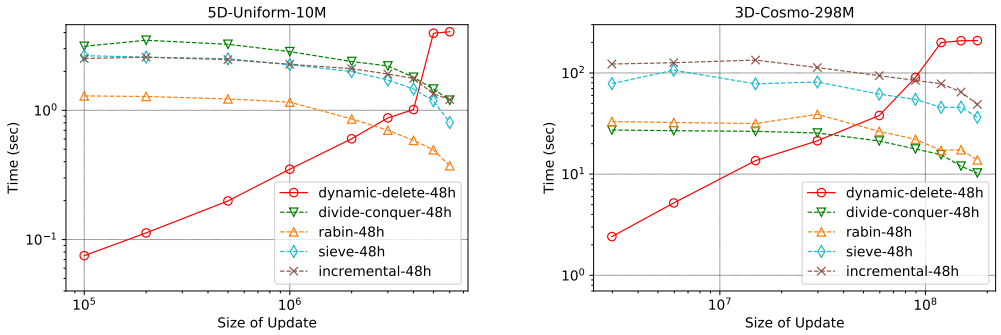


Figure 5 Plots of running time (in seconds) vs. deletion batch size for the dynamic and static methods using 48 cores with hyper-threading on 5D-Uniform-10M and 3D-Cosmo-298M. The plots are in log-log scale.

batch is smaller than 10 million. In general, both the static and dynamic algorithms require more time to process the updates when the batch size is larger. The dynamic algorithm is much more advantageous for small to moderate batch sizes.

Efficiency of Batch Deletions. We evaluate the performance of dynamic batch deletion vs. using a static algorithm to recompute the closest pair. In this experiment, we are given the closest pair of all n points in the data set, and perform a deletion of b points. We compare the time taken for the dynamic algorithm to process one batch deletion of size b, vs. that of a static algorithm for recomputing the closest pair for the $n - b$ remaining points. Figure 5 shows the running time vs. deletion batch size for 5D-Uniform-10M and 3D-Cosmo-298M. For 5D-Uniform-10M, the dynamic algorithm outperforms the fastest static algorithm when the batch size is less than 3 million. For 3D-Cosmo-298M, the dynamic algorithm outperforms the static algorithm when the batch size is less than 60 million. In general, our dynamic algorithm requires more time to process the update when the batch size is larger, while the converse is true for the static algorithms.

Compared to the fastest static algorithms on our data sets, we find that it is faster to use our dynamic algorithm for batch sizes of up to 20% of the data set.

Static Methods. We evaluate and compare the static algorithms and present all detailed running times in Table 1. Among the four parallel static algorithms, Rabin's algorithm is on average 7.63x faster than the rest of the algorithms across all data sets. The divide-and-conquer, sieve, and the incremental algorithms are on average 17.86x, 2.29x, and 2.73x slower than Rabin's algorithm, respectively. The divide-and-conquer algorithm actually achieves the fastest parallel running time on 7 out of the 11 data sets. However, it is significantly slower for most of the higher dimensional data sets, due to its higher complexity with increased dimensionality. The sieve algorithm and the incremental algorithm, though doing the same amount of work in theory as Rabin's algorithm, have higher constant factor overheads.

Parallel Speedup and Work-Efficiency. We measure the parallel speedups of our implementations by dividing the 1-thread time by the 48-core with hyper-threading time. Our parallel batch-dynamic algorithm achieves up to 38.57x self-relative speedup (15.10x on average across all batch sizes), averaging over both insertions and deletions. Our static implementations achieve up to 51.45x speedup (29.42x on average). Specifically, the divide-and-conquer

Table 1 Running times (in seconds) of static algorithms. "Seq" denotes the sequential implementation. "1t" and "48h" denote the parallel implementation run on 1 thread and 48 cores with hyper-threading, respectively.

	Divide-Conquer			Rabin			Sieve			Incremental		
	Seq	1t	48h	Seq	1t	48h	Seq	1t	48h	Seq	1t	48h
2D-Uniform-10M	9.54	9.62	0.24	11.2	11.6	0.28	23.3	24.5	0.81	22.1	17.7	1.02
3D-Uniform-10M	24.9	25.2	0.66	28.4	30.5	0.78	60.3	60.6	1.82	50.5	46.2	2.50
5D-Uniform-10M	101	136	3.04	25.3	28.4	1.28	56.7	60.6	2.63	49.2	50.3	2.40
7D-Uniform-10M	561	618	14.7	81.7	82.8	1.70	124	135	4.24	93.7	106	4.58
2D-SS-varden-10M	7.58	8.95	0.23	10.5	11.2	0.26	22.2	22.8	0.94	23.4	17.5	1.11
3D-SS-varden-10M	17.3	19.1	0.51	28.4	29.1	0.77	58.4	58.3	1.68	48.7	43.1	1.97
5D-SS-varden-10M	24.9	33.4	0.82	22.6	26.1	1.43	47.2	49.3	2.58	40.4	41.7	2.44
7D-SS-varden-10M	43.1	50.3	1.33	33.1	34.0	1.61	64.4	70.9	3.00	43.4	48.0	2.53
7D-Household-2M	342	392	13.4	7.23	7.70	0.40	15.9	18.1	0.73	13.8	15.7	0.94
16D-Chem-4M	315	499	202	38.3	39.8	1.38	88.2	96.7	2.68	59.1	70.8	3.91
3D-Cosmo-298M	750	747	20.7	1243	1625	31.6	3383	2819	70.6	3456	2629	104

algorithm, Rabin's algorithm, the sieve algorithm, and the incremental algorithm achieve average self-relative speedups of 35.17x, 33.84x, 29.22x, and 19.45x, respectively; in addition, they achieve an average speedup of 19.10x, 23.56x, 10.56x, and 9.23x, respectively, over the fastest serial algorithm for each data set.

Our parallel implementations when run on one thread demonstrate modest overheads over their sequential counterparts. Our parallel batch-dynamic algorithm running on 1 thread has only 1.13x lower throughput on average over our sequential implementation of the algorithm. For the static algorithms, the parallel divide-and-conquer, Rabin's, sieve, and incremental algorithms running on 1 thread are only 1.18x, 1.08x, 1.04x, and 1.00x slower on average, respectively, than their corresponding sequential algorithms.

References

1 Chem dataset. URL: `https://archive.ics.uci.edu/ml/datasets/Gas+sensor+array+under+dynamic+gas+mixtures`.

2 Pankaj K. Agarwal, Haim Kaplan, and Micha Sharir. Kinetic and dynamic data structures for closest pair and all nearest neighbors. *ACM Transactions on Algorithms (TALG)*, 5(1):1–37, 2008.

3 Suguru Arimoto and Hiroshi Noborio. A 3d closest pair algorithm and its applications to robot motion planning. *IFAC Proceedings Volumes*, 21(16):471–480, 1988.

4 Mikhail J. Atallah and Michael T. Goodrich. Efficient parallel solutions to some geometric problems. *J. Parallel Distrib. Comput.*, 3(4):492–507, 1986.

5 Bahareh Banyassady and Wolfgang Mulzer. A simple analysis of Rabin's algorithm for finding closest pairs. *European Workshop on Computational Geometry (EuroCG)*, 2007.

6 Jon L. Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, 1975.

7 Jon L. Bentley. Multidimensional divide-and-conquer. *Commun. ACM*, 23(4):214–229, 1980.

8 Jon L. Bentley and Michael I. Shamos. Divide-and-conquer in multidimensional space. In *ACM Symposium on Theory of Computing (STOC)*, pages 220–230, 1976.

9 Sergei N. Bespamyatnikh. An optimal algorithm for closest-pair maintenance. *Discrete & Computational Geometry*, 19(2):175–195, 1998.

10 Guy E. Blelloch, Yan Gu, Julian Shun, and Yihan Sun. Parallelism in randomized incremental algorithms. In *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, page 467–478, 2016.

11 Guy E. Blelloch and Bruce M. Maggs. Parallel algorithms. In *Algorithms and Theory of Computation Handbook: Special Topics and Techniques*, pages 25–25. Chapman & Hall/CRC, 2010.

12 Richard P. Brent. The parallel evaluation of general arithmetic expressions. *J. ACM*, 21(2):201–206, 1974.

13 Paul B. Callahan and S. Rao Kosaraju. Algorithms for dynamic closest pair and n-body potential fields. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, page 263–272, 1995.

14 Timothy M. Chan. Geometric applications of a randomized optimization technique. *Discrete & Computational Geometry*, 22(4):547–567, 1999.

15 Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms (3rd edition)*. MIT Press, 2009.

16 Martin Dietzfelbinger, Torben Hagerup, Jyrki Katajainen, and Martti Penttonen. A reliable randomized algorithm for the closest-pair problem. *J. Algorithms*, 25(1):19–51, 1997.

17 Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL: `http://archive.ics.uci.edu/ml`.

18 David Eppstein. Fast hierarchical clustering and other applications of dynamic closest pairs. *J. Experimental Algorithmics*, 5:1–es, 2000.

19 Jordi Fonollosa, Sadique Sheik, Ramón Huerta, and Santiago Marco. Reservoir computing compensates slow response of chemosensor arrays exposed to fast varying gas concentrations in continuous monitoring. *Sensors and Actuators B: Chemical*, 215:618–629, 2015.

20 Steve Fortune and John Hopcroft. A note on rabin's nearest-neighbor algorithm. *Information Processing Letters*, 8(1):20–23, 1979.

21 Junhao Gan and Yufei Tao. On the hardness and approximation of euclidean DBSCAN. *ACM Transactions Database Systems*, 42(3):14:1–14:45, 2017.

22 Joseph Gil, Yossi Matias, and Uzi Vishkin. Towards a theory of nearly constant time parallel algorithms. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 698–710, 1991.

23 Mordecai Golin, Rajeev Raman, Christian Schwarz, and Michiel Smid. Simple randomized algorithms for closest pair problems. *Nordic J. of Computing*, 2(1):3–27, 1995.

24 Mordecai Golin, Rajeev Raman, Christian Schwarz, and Michiel Smid. Randomized data structures for the dynamic closest-pair problem. *SIAM J. Scientific Computing*, 27(4):1036–1072, 1998.

25 Klaus Hinrichs, Jurg Nievergelt, and Peter Schorn. Plane-sweep solves the closest pair problem elegantly. *Information Processing Letters*, 26(5):255–261, 1988.

26 Joseph JaJa. *Introduction to Parallel Algorithms*. Addison-Wesley Professional, 1992.

27 Sanjiv Kapoor and Michiel Smid. New techniques for exact and approximate dynamic closest-point problems. *SIAM J. Scientific Computing*, 25(4):775–796, 1996.

28 Samir Khuller and Yossi Matias. A simple randomized sieve algorithm for the closest-pair problem. *Information and Computation*, 118(1):34–37, April 1995.

29 YongChul Kwon, Dylan Nunley, Jeffrey P. Gardner, Magdalena Balazinska, Bill Howe, and Sarah Loebman. Scalable clustering algorithm for N-body simulations in a shared-nothing cluster. In *Scientific and Statistical Database Management*, pages 132–150, 2010.

30 Md. Nasir Uddin Laskar and TaeChoong Chung. Mobile robot path planning : an efficient distance computation between obstacles using discrete boundary model (dbm), 2012.

31 Charles E. Leiserson. The Cilk++ concurrency platform. *J. Supercomputing*, 51(3), 2010.

32 Hans-Peter Lenhof and Michiel Smid. Sequential and parallel algorithms for the k closest pairs problem. *International J. of Computational Geometry & Applications*, 5(03):273–288, 1995.

33 Philip D. MacKenzie and Quentin F. Stout. Ultrafast expected time parallel algorithms. *J. Algorithms*, 26(1):1–33, 1998.

34 Mark H. Overmars. Dynamization of order decomposable set problems. *J. Algorithms*, 2(3):245–260, 1981.

35 Maria Cristina Pinotti and Geppino Pucci. Parallel algorithms for priority queue operations. *Theoretical Computer Science (TCS)*, 148(1):171–180, 1995.

36 Michael O. Rabin. Probabilistic algorithms, 1976.

37 Sanguthevar Rajasekaran and Sudipta Pathak. Efficient algorithms for the closest pair problem and applications. *arXiv preprint*, 2014. `arXiv:1407.5609`.

38 Sanguthevar Rajasekaran and John H. Reif. Optimal and sublogarithmic time randomized parallel sorting algorithms. *SIAM J. Scientific Computing*, 18(3):594–607, 1989.

39 Christian Schwarz and Michiel Smid. An $O(n \log n \log \log n)$ algorithm for the on-line closest pair problem. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, SODA '92, page 280–285, USA, 1992. Society for Industrial and Applied Mathematics.

40 Christian Schwarz, Michiel Smid, and Jack Snoeyink. An optimal algorithm for the on-line closest-pair problem. *Algorithmica*, 12(1):18–29, 1994.

41 Michael I. Shamos and Dan Hoey. Closest-point problems. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 151–162, 1975.

42 Michiel Smid. Maintaining the minimal distance of a point set in less than linear time. In *Algorithms Rev.*, pages 33–44, 1991.

43 Michiel Smid. Maintaining the minimal distance of a point set in polylogarithmic time. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1–6, 1991.

44 Michiel Smid. Closest-point problems in computational geometry. In *Handbook of Computational Geometry*, pages 877–935. North Holland / Elsevier, 2000.

45 Yihan Sun and Guy E. Blelloch. Parallel range, segment and rectangle queries with augmented maps. In *Algorithm Engineering and Experiments (ALENEX)*, pages 159–173, 2019.

46 Kenneth J. Supowit. New techniques for some dynamic closest-point and farthest-point problems. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 84–90, 1990.

47 Uzi Vishkin. Thinking in parallel: Some basic data-parallel algorithms. University of Maryland, 2010.

An Interactive Tool for Experimenting with Bounded-Degree Plane Geometric Spanners

Fred Anderson
School of Computing, University of North Florida, Jacksonville, FL, USA

Anirban Ghosh
School of Computing, University of North Florida, Jacksonville, FL, USA

Matthew Graham
School of Computing, University of North Florida, Jacksonville, FL, USA

Lucas Mougeot
School of Computing, University of North Florida, Jacksonville, FL, USA

David Wisnosky
School of Computing, University of North Florida, Jacksonville, FL, USA

Abstract

The construction of bounded-degree plane geometric spanners has been a focus of interest in the field of geometric spanners for a long time. To date, several algorithms have been designed with various trade-offs in degree and stretch factor. Using JSXGraph, a state-of-the-art JavaScript library for geometry, we have implemented seven of these sophisticated algorithms so that they can be used for further research and teaching computational geometry. We believe that our interactive tool can be used by researchers from related fields to understand and apply the algorithms in their research. Our tool can be run in any modern browser. The tool will be permanently maintained by the second author at `https://ghoshanirban.github.io/bounded-degree-plane-spanners/index.html`

2012 ACM Subject Classification Theory of computation → Sparsification and spanners

Keywords and phrases graph approximation, Delaunay triangulations, geometric spanners, plane spanners, bounded-degree spanners

Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.61

Category Media Exposition

Supplementary Material *Software (Tool)*:
`https://ghoshanirban.github.io/bounded-degree-plane-spanners/index.html`

Funding Research on this paper is supported by the NSF Award CCF-1947887 and by the University of North Florida Academic Technology Grant.

1 Introduction

Given a set P of n points in the Euclidean plane, a *geometric t-spanner* on P is a geometric graph $G := (P, E)$, such that for every pair of points $u, v \in P$, the distance between them in G (the length of a shortest path between u, v in G) is at most t times their Euclidean distance $|uv|$, for some $t \geq 1$. The complete geometric graph on P is a 1-spanner with $\Theta(n^2)$ edges. The quantity t is referred to as the *stretch factor* of G. A geometric spanner G is *plane* if it is crossing-free. If there is no necessity to specify t, we simply use the term *geometric spanner*.

Bose, Gudmundsson, and Smid [7] were the first to show that there always exists a plane 8.3-spanner of degree at most 27 on any point set. This result was subsequently improved in a series of papers [8, 3, 10, 6, 25, 22] in terms of degree and stretch factor. Bonichon et al. [5] reduced the degree to 4 with $t \approx 156.8$. Soon after this, Kanj et al. improved this stretch factor upper bound to 20 in [19]. A summary of these results is presented in Table 1.

Table 1 A summary of results on constructions of plane geometric spanners, sorted by the degree (Δ) they guarantee. The implemented algorithms are marked in bold. The best known upper bound of 1.998 for the stretch factor of the L_2-Delaunay triangulation [27] is used in this table for expressing the stretch factors. These algorithms run in time that is polynomial in n.

Reference	Δ	Upper bound on stretch factor
Bose, Gudmundsson, and Smid [7]	27	$1.998(\pi + 1) \approx 8.3$
Li and Wang [22]	23	$1.998(1 + \frac{\pi}{\sqrt{2}}) \approx 6.4$
Bose, Smid, and Xu [10]	17	$1.998(2 + 2\sqrt{3} + \frac{3\pi}{2} + 2\pi \sin\frac{\pi}{12}) \approx 23.6$
Kanj, Perković, and Xia [20]	14	$1.998(1 + \frac{2\pi}{14\cos(\pi/14)}) \approx 2.9$
Bose, Hill, and Smid [8]	8	$1.998\left(1 + \frac{2\pi}{6\cos(\pi/6)}\right) \approx 4.4$
Bose, Carmi, and Chaitman-Yerushalmi [6]	7	$1.998(1 + \sqrt{2})^2 \approx 11.6$
Bose, Carmi, and Chaitman-Yerushalmi [6]	6	$1.998\left(\frac{1}{1-\tan(\pi/7)(1+1/\cos(\pi/14))}\right) \approx 81.7$
Bonichon et al. [3]	6	6
Bonichon et al. [5]	4	$\sqrt{4 + 2\sqrt{2}}(19 + 29\sqrt{2}) \approx 156.8$
Kanj, Perković, and Türkoğlu [19]	4	20

The question whether the degree can be reduced to 3 keeping t bounded, still remains open at this time; refer to [9, Problem 14] and [26, Chapter 32]. If one does not insist on constructing a plane geometric spanner, Das and Heffernan [14] showed that degree 3 is always achievable. It is shown in the book [24, Section 20.1] by Narasimhan and Smid that no degree-2 plane spanner of the infinite integer lattice can have constant stretch factor. Thus, a minimum degree of 3 is necessary to achieve a constant stretch factor. Biniaz et al. [2] showed that if P is convex, then it is always possible to construct a plane 5.2-spanner having degree 3. From the other direction, lower bounds on the stretch factors of plane spanners for finite point sets have been investigated in [16, 15, 21, 23]. Plane geometric spanners find their applications in robotics and wireless networks where edge crossings may cause interference. An advantage of using plane spanners is that they have $O(n)$ edges and consequently take less storage space. In related works, the construction of plane hop spanners (where the number of hops in shortest paths is of interest) for unit disk graphs has been considered in [11, 1, 17].

Every algorithm designed so far that can construct bounded-degree plane spanners relies on some variant of Delaunay triangulation as the starting point. The rationale behind this is that these triangulations are geometric spanners [27, 12, 13, 4] and are plane by definition. As such, this family of plane spanner construction algorithms has turned out to be a fascinating application of Delaunay triangulation. In this work, we have implemented a set of seven novel algorithms that rely only on the L_2-Delaunay triangulation; refer to Table 1. The algorithms in Table 1 that are not implemented, use other kinds of Delaunay triangulations and are not considered in this work due to their inherent implementation complexities. To our knowledge, this is the first time that these novel algorithms have been implemented. We urge the readers to refer to the source papers to gain an understanding of the implemented algorithms.

Our implementations have two-fold contributions. First, they will help in the research of geometric spanners where tools are rarely available for experiments. Second, they can be used in teaching geometric spanners in computational geometry courses. This tool will be

permanently maintained by the second author in his GitHub[1]. In-browser implementations of path-greedy, gap-greedy, Θ-graph, and Yao-graph algorithms were considered by Farshi and Hosseini in [18].

2 Implementation and usage

We have implemented the spanner algorithms using the JSXGraph library for attractive visualization and easy interaction. A technical description and documentation of this tool is included in the tool itself; click on the `ReadMe` button in the tool to launch this. We encourage users to read this read-me before using the tool.

After the tool is launched in a browser, the user can enter points manually either by clicking on the canvas or by entering coordinates explicitly in a text-box. The tool also comes with several built-in point sets which can also be used for experiments along with the manually entered points. Random point generation is also supported by the tool. Once the point set is finalized, an algorithm needs to be selected for spanner construction. Some of these algorithms accepts an extra parameter from the user for spanner construction which can be easily specified using a slider.

The tool draws the generated spanner and outputs the following: $|V|$, $|E|$, the exact stretch factor of G, degree of G, average degree (taken over all points in P), lightness (the ratio of the weight of G to that of the Euclidean Minimum Spanning Tree on P), and the spanner edges. It also shows the point pair that achieves the exact stretch factor for G in red, along with a shortest path between them in G. The built-in screenshot support allows the user to export the current board to a `png` or `svg` image for future uses.

3 Conclusions

We believe that our tool can bring new insights to the research of geometric spanners. In particular, we hope that this tool will aid researchers to solve the fascinating open problem posed in [9, Problem 14] and [26, Chapter 32] that asks whether degree-3 plane geometric spanners having bounded stretch factor are always possible. This tool can also be used in teaching computational geometry courses where geometric spanners hold special importance. Furthermore, researchers from related fields such as robotics and networking can use our tool in their research.

References

1 Ahmad Biniaz. Plane hop spanners for unit disk graphs: Simpler and better. *Comput. Geom.*, 89:101622, 2020. doi:10.1016/j.comgeo.2020.101622.

2 Ahmad Biniaz, Prosenjit Bose, Jean-Lou De Carufel, Cyril Gavoille, Anil Maheshwari, and Michiel Smid. Towards plane spanners of degree 3. *Journal of Computational Geometry*, 8(1):11–31, 2017.

3 Nicolas Bonichon, Cyril Gavoille, Nicolas Hanusse, and Ljubomir Perković. Plane spanners of maximum degree six. In *International Colloquium on Automata, Languages, and Programming*, pages 19–30. Springer, 2010.

4 Nicolas Bonichon, Cyril Gavoille, Nicolas Hanusse, and Ljubomir Perković. The stretch factor of l_1-and l_∞-delaunay triangulations. In *European Symposium on Algorithms*, pages 205–216. Springer, 2012.

5 Nicolas Bonichon, Iyad Kanj, Ljubomir Perković, and Ge Xia. There are plane spanners of degree 4 and moderate stretch factor. *Discrete & Computational Geometry*, 53(3):514–546, 2015.

[1] URL to the tool: `https://ghoshanirban.github.io/bounded-degree-plane-spanners/index.html`

6 Prosenjit Bose, Paz Carmi, and Lilach Chaitman-Yerushalmi. On bounded degree plane strong geometric spanners. *Journal of Discrete Algorithms*, 15:16–31, 2012.

7 Prosenjit Bose, Joachim Gudmundsson, and Michiel Smid. Constructing plane spanners of bounded degree and low weight. *Algorithmica*, 42(3-4):249–264, 2005.

8 Prosenjit Bose, Darryl Hill, and Michiel Smid. Improved spanning ratio for low degree plane spanners. *Algorithmica*, 80(3):935–976, 2018.

9 Prosenjit Bose and Michiel Smid. On plane geometric spanners: A survey and open problems. *Computational Geometry*, 46(7):818–830, 2013.

10 Prosenjit Bose, Michiel Smid, and Daming Xu. Delaunay and diamond triangulations contain spanners of bounded degree. *International Journal of Computational Geometry & Applications*, 19(02):119–140, 2009.

11 Nicolas Catusse, Victor Chepoi, and Yann Vaxès. Planar hop spanners for unit disk graphs. In *International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics*, pages 16–30. Springer, 2010.

12 L Paul Chew. There are planar graphs almost as good as the complete graph. *Journal of Computer and System Sciences*, 39(2):205–219, 1989.

13 LP Chew. There is a planar graph almost as good as the complete graph, 2nd symp. on computational geometry, 1986.

14 Gautam Das and Paul J Heffernan. Constructing degree-3 spanners with other sparseness properties. *International Journal of Foundations of Computer Science*, 7(02):121–135, 1996.

15 Adrian Dumitrescu and Anirban Ghosh. Lattice spanners of low degree. *Discrete Mathematics, Algorithms and Applications*, 8(03):1650051, 2016.

16 Adrian Dumitrescu and Anirban Ghosh. Lower bounds on the dilation of plane spanners. *International Journal of Computational Geometry & Applications*, 26(02):89–110, 2016.

17 Adrian Dumitrescu, Anirban Ghosh, and Csaba D Tóth. Sparse hop spanners for unit disk graphs. In *31st International Symposium on Algorithms and Computation (ISAAC 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

18 Mohammad Farshi and Seyed Hossein Hosseini. Visualization of geometric spanner algorithms. In *32nd International Symposium on Computational Geometry (SoCG 2016)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

19 Iyad Kanj, Ljubomir Perkovic, and Duru Türkoğlu. Degree four plane spanners: Simpler and better. *Journal of Computational Geometry*, 8(2):3–31, 2017.

20 Iyad A Kanj, Ljubomir Perković, and Ge Xia. On spanners and lightweight spanners of geometric graphs. *SIAM Journal on Computing*, 39(6):2132–2161, 2010.

21 Rolf Klein, Martin Kutz, and Rainer Penninger. Most finite point sets in the plane have dilation > 1. *Discrete & Computational Geometry*, 53(1):80–106, 2015.

22 Xiang-Yang Li and Yu Wang. Efficient construction of low weighted bounded degree planar spanner. *International Journal of Computational Geometry & Applications*, 14(01n02):69–84, 2004.

23 Wolfgang Mulzer. Minimum dilation triangulations for the regular n-gon. *Master's thesis, Freie Universität Berlin, Germany*, 2004.

24 Giri Narasimhan and Michiel Smid. *Geometric spanner networks*. Cambridge University Press, 2007.

25 Ljubomir Perkovic and Iyad A Kanj. On geometric spanners of euclidean and unit disk graphs. In *25th International Symposium on Theoretical Aspects of Computer Science*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2008.

26 Csaba D Toth, Joseph O'Rourke, and Jacob E Goodman. *Handbook of discrete and computational geometry*. Chapman and Hall/CRC, 2017.

27 Ge Xia. The stretch factor of the delaunay triangulation is less than 1.998. *SIAM Journal on Computing*, 42(4):1620–1659, 2013.

Can You Walk This? Eulerian Tours and IDEA Instructions

Aaron T. Becker
Cullen College of Engineering, University of Houston, TX, USA

Sándor P. Fekete
Department of Computer Science, TU Braunschweig, Germany

Matthias Konitzny
Department of Computer Science, TU Braunschweig, Germany

Sebastian Morr
Department of Computer Science, TU Braunschweig, Germany

Arne Schmidt
Department of Computer Science, TU Braunschweig, Germany

Abstract

We illustrate and animate the classic problem of deciding whether a given graph has an Eulerian path. Starting with a collection of instances of increasing difficulty, we present a set of pictorial instructions, and show how they can be used to solve all instances. These *IDEA instructions* ("A series of nonverbal algorithm assembly instructions") have proven to be both entertaining for experts and enlightening for novices. We (w)rap up with a song and dance to Euler's original instance.

2012 ACM Subject Classification Mathematics of computing → Discrete mathematics; Applied computing → Education

Keywords and phrases Eulerian tours, algorithms, education, IDEA instructions

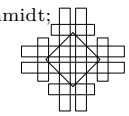
Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.62

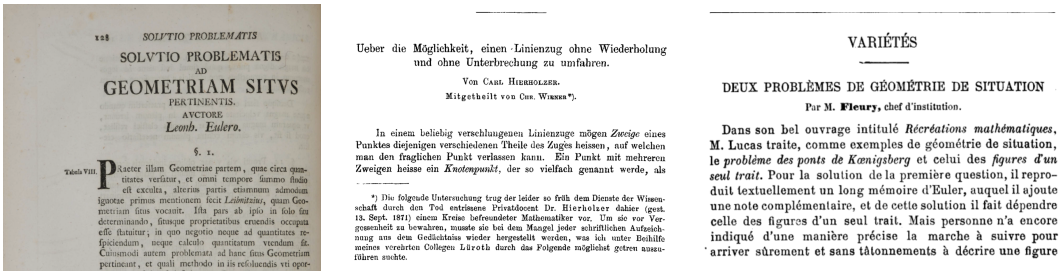
Category Media Exposition

1 Introduction

Deciding whether a connected graph $G = (V, E)$ has an Eulerian path is a natural problem of graph theory: Find a path P that contains all edges in E, starting at a suitable vertex s and ending at a vertex t. As a path, P is connected and may not contain any edge more than once; therefore, following an Eulerian path corresponds to drawing all of the edges in one continuous stroke, without duplicating any edge or lifting the pen – or crossing all bridges in a city in one contiguous walk without duplicating a bridge. First introduced by Euler in 1741 [2], it is not just a classic, but arguably *the* problem that started graph theory itself.

The algorithmic side of the problem spans several centuries, with publications in a variety of languages; see Fig. 1. In his paper (published in Latin), Euler gave a necessary condition for the existence of P: there may be at most two vertices of odd degree, which would have to be s and t. Hierholzer [6] (in an 1873 posthumous paper published in German) gave an algorithmic proof that this condition is sufficient, based on iteratively merging cycles. An alternative was given by Fleury [4] in 1883 (published in French): If the necessary condition is satisfied, it is possible to find an Eulerian path by greedily following edges, starting at an odd-degree vertex s if one exists, subject to never disconnecting the set of unused edges. From an algorithmic point of view, Fleury's algorithm is somewhat inefficient, as it requires keeping track of connected components; from an intuitive perspective, Fleury's method is quite elegant, as it does indeed provide a method for drawing the graph in one stroke, without resorting to retroactively including leftover cycles according to Hierholzer.



SOLVTIO PROBLEMATIS

SOLVTIO PROBLEMATIS AD GEOMETRIAM SITVS PERTINENTIS.

AVCTORE *Leonh. Eulero.*

§. 1.

Tabula VIII. Praeter illam Geometriae partem, quae circa quantitates versatur, et omni tempore summo studio est exculta, alterius partis etiamnum admodum ignotae primus mentionem fecit *Leibnitius*, quam Geometriam situs vocauit. Ita pars ab ipso in solo situ determinando, situsque proprietatibus eruendis occupata esse statuitur; in quo negotio neque ad quantitates respiciendum, neque calculo quantitatum vtendum sit. Cuiusmodi autem problemata ad hanc situs Geometriam pertineant, et quali methodo in iis resoluendis vti opor-

Ueber die Möglichkeit, einen Linienzug ohne Wiederholung und ohne Unterbrechung zu umfahren.

Von Carl Hierholzer.

Mitgetheilt von Chr. Wiener*).

In einem beliebig verschlungenen Linienzuge mögen *Zweige* eines Punktes diejenigen verschiedenen Theile des Zuges heissen, auf welchen man den fraglichen Punkt verlassen kann. Ein Punkt mit mehreren Zweigen heisse ein *Knotenpunkt*, der so vielfach genannt werde, als

*) Die folgende Untersuchung trug der leider so früh dem Dienste der Wissenschaft durch den Tod entrissene Privatdocent Dr. Hierholzer dahier (gest. 13. Sept. 1871) einem Kreise befreundeter Mathematiker vor. Um sie vor Vergesseuheit zu bewahren, musste sie bei dem Mangel jeder schriftlichen Aufzeichnung aus dem Gedächtniss wieder hergestellt werden, was ich unter Beihilfe meines verehrten Collegen Lüroth durch das Folgende möglichst getreu auszuführen suchte.

VARIÉTÉS

DEUX PROBLÈMES DE GÉOMÉTRIE DE SITUATION

Par M. **Fleury**, chef d'institution.

Dans son bel ouvrage intitulé *Récréations mathématiques*, M. Lucas traite, comme exemples de géométrie de situation, le *problème des ponts de Kœnigsberg* et celui des *figures d'un seul trait*. Pour la solution de la première question, il reproduit textuellement un long mémoire d'Euler, auquel il ajoute une note complémentaire, et de cette solution il fait dépendre celle des figures d'un seul trait. Mais personne n'a encore indiqué d'une manière précise la marche à suivre pour arriver sûrement et sans tâtonnements à décrire une figure

Figure 1 Three fundamental papers on the topic that describe the relationship between geometry and graph theory in three different languages: (Left) Euler [2], (Center) Hierholzer [6], (Right) Fleury [4].

2 Eulerian paths and education

Euler made use of the combination of a specific instance and a relatively accessible problem to develop the universal and deep concept of graphs: He provided a very concrete example, instead of a series of axioms and abstract definitions. This makes questions of Eulerian paths ideally suited for introducing novices to both graphs and algorithmic problems. Not only first-year undergraduates, but also high-school and even elementary school students quickly grasp the underlying concepts when challenged with a sequence of instances of increasing difficulty. Progressing from easier to harder instances ensures a good balance between pitfalls and rewards, leading to insights and generalizations, providing both motivation, a sense for solutions and difficulties, and appreciation for general methods and algorithms. Such a set of instances was used in actual work with students at all levels, starting at elementary school. The first set of twelve instances (Figure 2) focuses on the role of odd vertices, supporting the discovery of their importance by gradually increasing the level of complexity. The second set (starting with (2.1), not shown in the figure, but used in the video) provides larger instances in which odd-degree vertices do not matter, but the role of connectivity is highlighted.

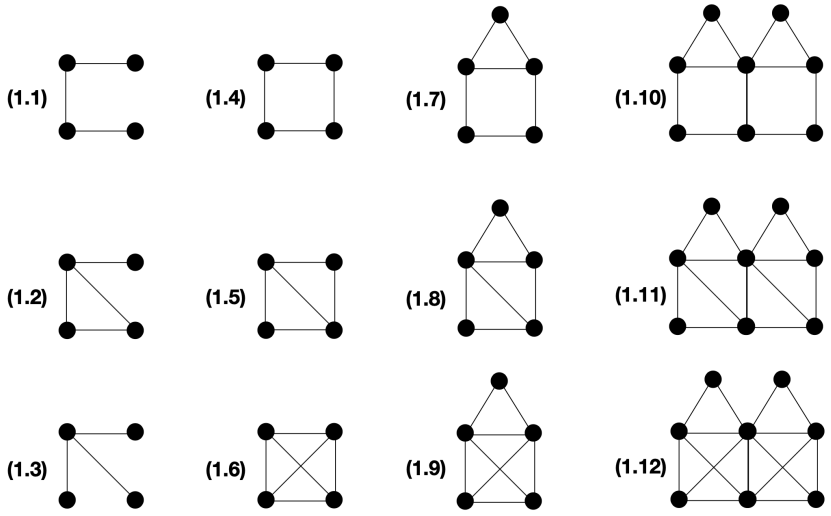


Figure 2 The set of instances used in the video.

3 IDEA instructions

One of the fundamental concepts of mathematics is abstraction: definitions, theorems and proofs ought to be valid regardless of a visual depiction. This angle is also quite natural and important when turning mathematical results into algorithms and computer code: "The computer doesn't need to understand the algorithm, its task is to run the program." (Tarjan [5]). However, the main purpose of a human brain (in particular, one in the process of being trained) is not to run code, but to develop an understanding for the underlying concepts and methods.

That is why two of us (Sándor Fekete and Sebastian Morr) have designed a series of instruction sheets that show algorithms without words, loosely inspired by furniture assembly instructions, but more challenging: While a piece of furniture is just a single, specific instance, an algorithm has to be ready for *any* instance. (See our website [3].) The absence of words forces the user to interpret the meaning of images, thereby enhancing the role of intuitive understanding. In combination with sets of instances, this encourages experimentation, so the IDEA instructions may also play the role of "cheat sheets" that provide light-bulb moments when trying to overcome a challenging difficulty. Experience shows that this strengthens both the interpretation of individual steps and the appreciation for crucial tricks.

For the specific problem of Eulerian paths, our instruction sheet (shown in Figure 3) provides a visual description of Fleury's algorithm. The setup proceeds in the usual algorithmic manner (given/wanted, input/output). The first step describes identifying the odd-degree vertices, while the second step instructs the user to mark them in the actual instance. Just like furniture instructions or mathematical structures, this comes with a minor puzzle for the user to figure out: Can there be instances with only a single odd-degree vertex? This has turned out to be a tantalizing question for curious students of all ages, and serves as a bridge (no pun intended) to further mathematical explorations.

The second part of the instruction sheet carries out the actual algorithm. This is split up between carrying out specific steps (left column) and figuring out a suitable next edge (right column). The latter is easily the most challenging step, in particular in a general, abstract manner, as it requires discovering the concepts of connected components and k-edge connectivity. However, given the context of bridges between islands, students are usually able to figure out this step with a combination of studying instances and referring to the instructions. This also highlights the seam between intuition and formalism, driving home the importance of clear definitions and setting the stage for universal notation and code: How can these picture puzzle steps be formalized when working towards actual implementations?

4 The Video

The video opens with the original problem in Königsberg, followed by a young boy presented with the problem set shown in Figure 2. After a sequence of successes and failures, the scene switches to the IDEA instruction sheet shown in Figure 3. This is applied to the set of instances, animating the algorithmic progress on each instance and the visual code. The video switches back to the young boy solving a large, exam-level problem for university students. The video concludes with a rap performance based on the following poem by Bill Tutte, under the pen name Blanche Descartes [1].

Some citizens of Königsberg
Were walking on the strand
Beside the river Pregel
With its seven bridges spanned.

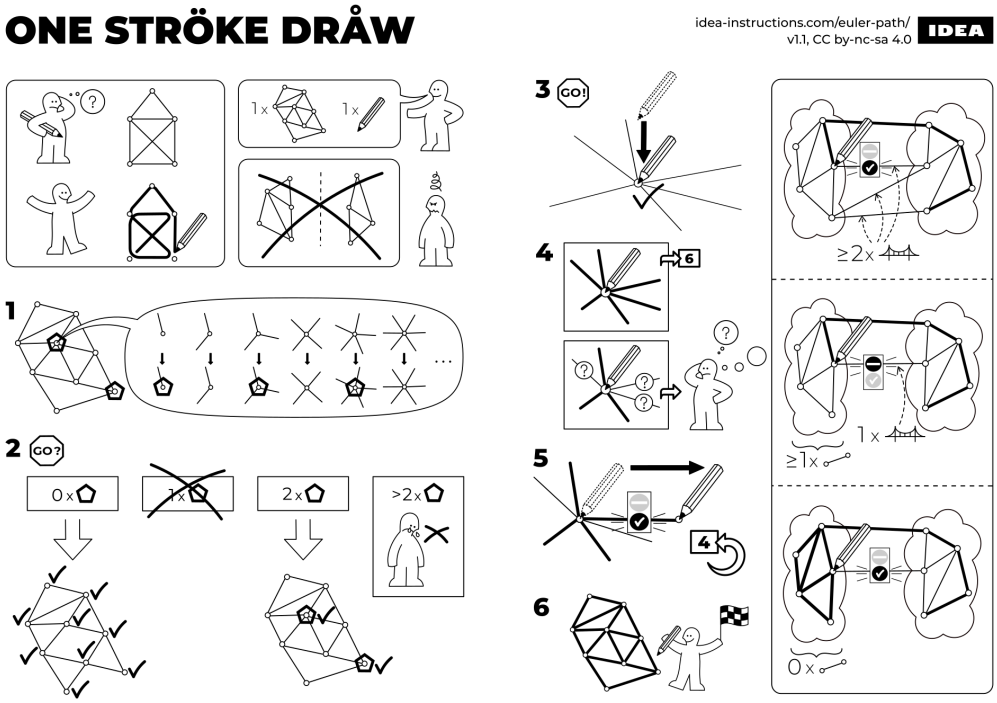


Figure 3 IDEA instructions for finding an Eulerian path: algorithmic description without words.

"O Euler, come and walk with us,"
Those burghers did beseech.
"We'll walk the seven bridges o'er,
And pass but once by each."

"It can't be done," thus Euler cried.
"Here comes the Q.E.D.
Your islands are but vertices
And four have odd degree."

References

1 Blanche Descartes. The expanding unicurse. In Frank Harari, editor, *Proof Techniques in Graph Theory*, page 25. Academic Press, 1969.

2 Leonhard Euler. Solutio problematis ad geometriam situs pertinentis. *Commentarii academiae scientiarum Petropolitanae*, pages 128–140, 1741.

3 Sándor P. Fekete and Sebastian Morr. IDEA instructions. `https://idea-instructions.com`, 2018.

4 M Fleury. Deux problèmes de géométrie de situation. *Journal de mathematiques éleméntaires*, 2(2):257–261, 1883.

5 Karen A Frenkel. An interview with the 1986 ACM Turing award recipients – John E. Hopcroft and Robert E. Tarjan. *Communications of the ACM*, 30(3):214–222, 1987.

6 Carl Hierholzer and Chr Wiener. Über die Möglichkeit, einen Linienzug ohne Wiederholung und ohne Unterbrechung zu umfahren. *Mathematische Annalen*, 6(1):30–32, 1873.

Shadoks Approach to Low-Makespan Coordinated Motion Planning

Loïc Crombez
Université Clermont-Auvergne and LIMOS, France

Guilherme D. da Fonseca
Aix Marseille Université and LIS, France

Yan Gerard
Université Clermont-Auvergne and LIMOS, France

Aldo Gonzalez-Lorenzo
Aix Marseille Université and LIS, France

Pascal Lafourcade
Université Clermont-Auvergne and LIMOS, France

Luc Libralesso
Université Clermont-Auvergne and LIMOS, France

Abstract

This paper describes the heuristics used by the `Shadoks`[1] team for the CG:SHOP 2021 challenge on motion planning. Using the heuristics outlined in this paper, our team won first place with the best solution to 202 out of 203 instances and optimal solutions to at least 105 of them.

2012 ACM Subject Classification Theory of computation → Computational geometry; Computing methodologies → Motion path planning

Keywords and phrases heuristics, motion planning, digital geometry, shortest path

Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.63

Category CG Challenge

Related Version *Full Version*: `https://arxiv.org/abs/2103.13956`

Supplementary Material
Audiovisual: `https://vimeo.com/515040997`
Software (Source Code): `https://github.com/gfonsecabr/shadoks-robots`
archived at `swh:1:dir:88b0688cdc71890d6582cdfc8e8c9c0e37b831b4`

Funding *Loïc Crombez*: This work has been sponsored by the French government research program "Investissements d'Avenir" through the IDEX-ISITE initiative 16-IDEX-0001 (CAP 20-25).
Guilherme D. da Fonseca: This work is supported by the French ANR PRC grant ADDS (ANR-19-CE48-0005).
Yan Gerard: This work is supported by the French ANR PRC grants ADDS (ANR-19-CE48-0005) and ACTIVmap (ANR-19-CE19-0005).
Pascal Lafourcade: This work is supported by the French ANR PRC grant MobiS5 (ANR-18-CE39-0019), DECRYPT (ANR-18-CE39-0007), and SEVERITAS (ANR-20-CE39-0005).
Luc Libralesso: This work is supported by the French ANR PRC grant DECRYPT (ANR-18-CE39-0007).

Acknowledgements We would like to thank Hélène Toussaint, Raphaël Amato, Boris Lonjon, and William Guyot-Lénat from LIMOS, as well as the Qarma and TALEP teams and Manuel Bertrand from LIS, who continue to make the computational resources of the LIMOS and LIS clusters available to our research. We would also like to thank the challenge organizers and other competitors for their time, feedback, and making this whole event possible.

[1] The team name comes from the animated series `https://en.wikipedia.org/wiki/Les_Shadoks`.

1 Introduction

We explain some heuristics used by the `Shadoks` team to solve the CG:SHOP 2021 challenge that considers a coordinated motion planning problem in the two-dimensional grid $\mathbb{Z}^2$. The goal is to move a set of n labeled unit squares called *robots* between given start and target grid cells without collisions. For more details, see the overview [3] and also [2].

The objective of the problem is to minimize the makespan[2] m. A trivial lower bound to the makespan is the largest obstacle-avoiding L_1 distance between the start and the target of a robot. Since the problem is symmetric with respect to the time, we may always exchange start-target positions and reverse the paths, keeping the best solution found.

The challenge CG:SHOP 2021 provided 203 instances containing between 10 and 9000 robots, out of which 202 of our solutions were the best ones among all the 17 teams who participated. To our surprise, we succeed in finding 105 solutions that match the trivial lower bound. Our strategy consists of two steps, presented in Section 2 and 3: finding a feasible solution and reducing its makespan.

At the beginning of the challenge, we tried some approaches from the multi-agent path finding literature [4, 7] (notably CBS [5]) to solve the challenge instances. To our surprise, they did not perform well. Indeed, the challenge instances are much denser than the ones in the literature. These instances are usually sparse in the number of agents (there are from 2 to 120 agents placed in grids with over 100,000 cells in these instances, where instances of the challenge contain hundreds or thousands of agents placed in grids never larger than 100×100). This structural difference has a dramatic effect on the performance of CBS and in our experiments, CBS fails to find solutions for most of the challenge instances (with the exception of some very small ones).

2 Initial Solutions

Feasibility is guaranteed for the challenge instances since the number of obstacles is finite and every start and target are located in the unbounded region of space. In this section, we show how to obtain feasible solutions with a moderate makespan. We divide the heuristics in two categories. In Section 2.1, we compute the solution one step at a time, considering multiple robots simultaneously. In Section 2.2, we compute the solutions one robot at a time.

The heuristics of the first category are not guaranteed to find a solution, but when they do they often find solutions of lower makespan than those of the second category. The algorithms of the second category are guaranteed to find a solution, but the resulting makespan may potentially be high.

2.1 Step by Step Computation

The problem of finding a solution for coordinated motion planning in a given number of steps can be modeled as an Integer Linear Problem (ILP) or equivalently as a SAT problem (see [8, 9] and references therein). While applying such an approach is intractable even for small instances, it can be adapted to find an initial solution. The general idea of the *Greedy* solver is to plan only a small number k of steps for the robots such that the overall distance to the targets decreases as much as possible and repeat until reaching the targets.

[2] The challenge also considered the objective of minimizing the sum of the distances, but we did not optimize our solutions for this version.

Our ILP model considers a Boolean variable $x_{r,P}$ for each robot r and for each possible path P of length k starting at the position of the robot. Constraints of having one and only one path per robot and avoiding obstacles and collisions between robots are easily expressed as linear inequalities. The objective function we maximize is the sum of all the variables with weight

$$\text{weight}(r, P) = \Big(\delta_r(p(0)) - \delta_r(p(k))\Big) \cdot \Big((\delta_r(p(0)))^2 + 1\Big),$$

where $p(0)$ and $p(k)$ are the first and the last positions of path P and $\delta_r(p)$ is the obstacle-avoiding distance from a point p to the target of robot r. The first factor encourages the solution to push the robots towards their targets, since it is better to get closer to target. The second factor prioritizes moving robots that are farther from their targets and we add one so that robots that are already at their target position are encouraged to remain there.

In practice, we set $k = 3$ and we only perform the first step of the planned moves so that the robots can *anticipate* the moves of the other robots. Using the CPLEX [1] library to solve these problems, we can handle instances with up to roughly 200 robots. Note that this Greedy algorithm is not guaranteed to find a solution, and it fails to solve instances with *corridors* such as `universe_bg_000`.

2.2 Robot by Robot Computation

The algorithms in this section compute the solution one robot at a time using an A* search. The search happens in 3-dimensional space where each robot state has integer coordinates of the form (x, y, t) for position coordinates x, y and time t. There are 5 possible *movements*, all of which increase t by one unit. One movement keeps the position x, y unchanged, while the other 4 movements increment or decrement one of the two coordinates. A movement is feasible if it does not violate any of the problem constraints, considering the current path of the other robots.

We refer to the *bounding box* as an integer axis-aligned rectangular region containing all the start, target positions and obstacles inside its strict interior (not on the boundary). Given a set of obstacles and a bounding box, the *depth* of a position p is the minimum obstacle-avoiding distance from p to a position outside the bounding box.

All algorithms in this section are based on a *storage network* N. A storage network is a set N of positions outside a predetermined bounding box such that for every position p in N, there exists a path that avoids all other positions of N and goes from p to some point in the bounding box. Each robot r_i is assigned to a distinct element of N, called the *storage* of r_i.

Initially, we set the path of each robot to be stationary at the start position. We sort the robots by *increasing start depth* and for each robot in order, we use A* search to find the shortest path from start to storage, replacing the previous stationary path. The order by which the robots are sorted guarantees that such a path exists.

After finding paths from start to storage for every robot, we proceed to the next phase of the algorithm. We now sort the robots by *decreasing target depth*. Again, the order of the robots guarantees that a path from storage to target exists. However, we do not compute such a path. Instead, we compute a path from start to target directly, whose existence is guaranteed by the existence of a path from start to storage and another one from storage to target. The following paragraphs describe the design of four different storage networks.

Cross. In the *Cross* strategy, we define the storage network N as the set of columns of even x coordinate lying directly above or below the bounding box and the set of rows of even y coordinate lying directly to the left or right of the bounding box, hence the name

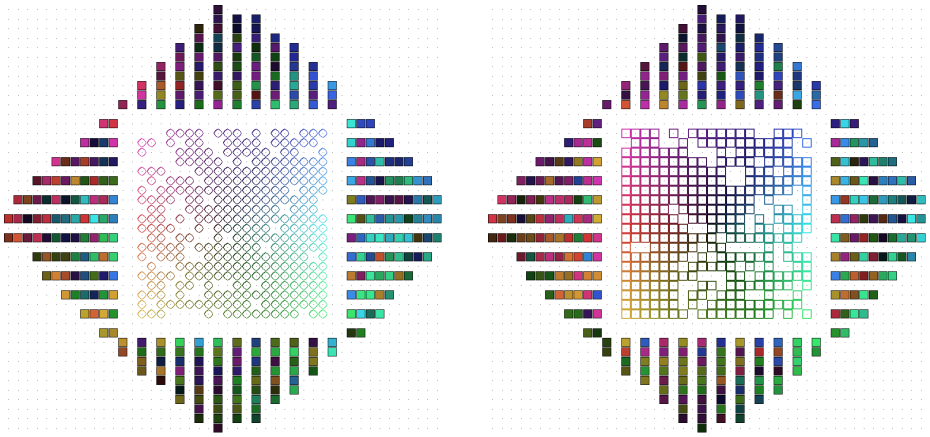


Figure 1 Cross storage network for the `small_free_016` instance colored based on start and target locations, respectively.

Cross. Then, we compute a maximal cardinality matching between the robots and N. We tried both minimum-weight matching and greedy matchings, minimizing a weight function that considers the distance from start to storage as well as the distance from storage to target. In the greedy matching version, robots are assigned a storage ordered by decreasing start-to-target distance. The result is represented in Figure 1.

Cootie Catcher. The previous strategy works very well for small or sparse instances. However, the different directions of the flow of robots from start to storage make the solutions inefficient for large dense instances. The *Cootie Catcher* strategy computes the storage using only the start location, in order to better exploit parallel movement of the robots. The storage network shape consisting of four diamonds is presented in Figure 2. For instances without obstacles, the strategy is guaranteed to find a path from start to storage using at most $w/2 + \mathcal{O}(1)$ steps, where w is the largest bounding box side. Surprisingly, this strategy also works well for many instances with obstacles.

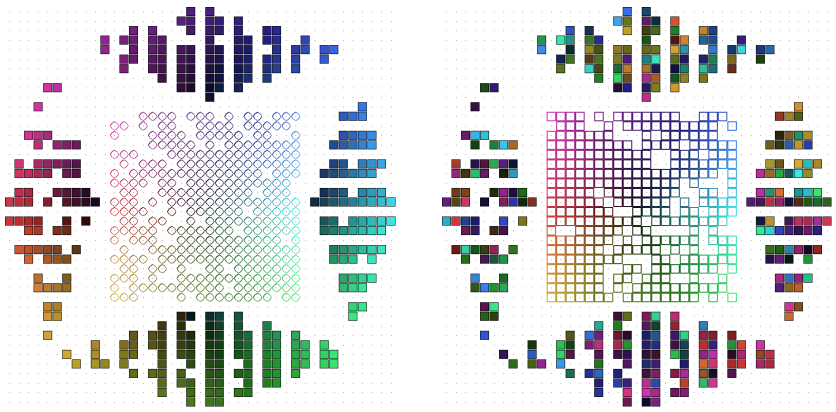


Figure 2 Cootie Catcher storage network for the `small_free_016` instance colored based on start and target locations, respectively.

Dichotomy. The weakness of the previous method is that robots may be assigned storage in a location that is opposite to the direction from start to target. In order to exploit parallel movements while taking the target location into consideration, we developed the *Dichotomy* strategy. The strategy only works for instances without obstacles.

We translate the coordinate system so that the origin is the center of the bounding box. The robots are partitioned into two sets called *left side* and *right side* according to the sign of the target location's x-coordinate. Left-side robots are assigned storage with positive x-coordinate while right-side robots are assigned storage with negative x-coordinate, as represented in Figure 3.

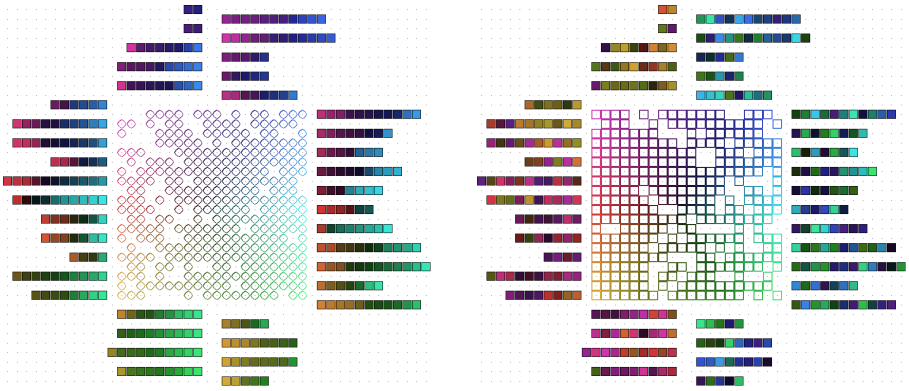


Figure 3 Dichotomy storage network for the `small_free_016` instance colored based on start and target locations, respectively.

Escape. This strategy focuses on instances with obstacles, especially on dense instances where the obstacles create bottlenecks. The goal of the *Escape* strategy is to clear the bounding box as quickly as possible. To do so, we move the robots by blocks as large as possible making efficient use of parallel movements. The Escape strategy defines layers inside the bounding box. Robots located in each layer will move in a straight line to reach the previous layer and then in another straight line outside of the bounding box as represented in Figure 4. We used a naive algorithm to define the layers, and partially redefined them by hand for the most complicated instances and the unsatisfying results.

3 Improving Solutions

In this section we discuss the two heuristics that we used to reduce the makespan of a given feasible solution. The first heuristic makes local changes to the solution, which remains feasible throughout the process, and possibly reduces the makespan. The second heuristic destroys the feasibility of the solution and either finds another solution of reduced makespan, or no feasible solution at all. Throughout, let m be the makespan of the input solution.

Feasible Optimizer. The idea of the *Feasible Optimizer* is the following. We iteratively remove the path of a robot r from the solution, and then use the A* algorithm to find a new (hopefully different) path for r. The A* algorithm may be tuned in several ways to produce different paths, and we do so in such a way that the makespan of the solution never increases and also that a robot is only allowed to move at time m if it already did so in the original path. This way, not only the makespan but also the number of robots moving at time m never increase. Next, we list some examples on how to modify the A* search.

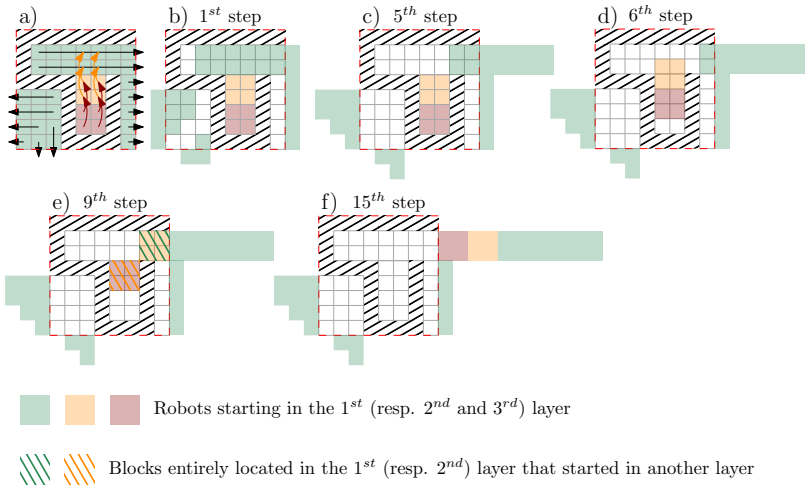


Figure 4 Escape strategy.

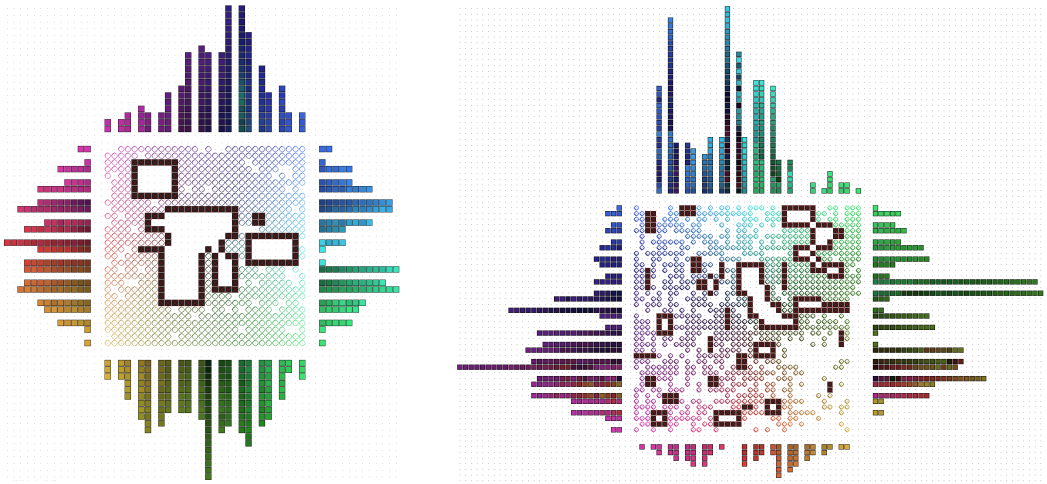


Figure 5 Escape storage network for the `medium_007` and `buffalo_003` instances, respectively.

- Find the path from start to target that reaches the target as quickly as possible but break ties using the sum of random weights given to each grid cell the robot passes through.
- Reversing the direction of time and then finding a path from target to start that reaches the start as quickly as possible. In the original time direction, that means that the robot will remain at the start for as long as possible.
- In the reversed case, force the robot to stay at target for a certain number of steps.

Conflict Optimizer. The previous optimization strategy may take very long to reduce the makespan. Next, we describe a more aggressive approach that leaves the feasible solution space and works far better than we expected. The algorithm uses a modified A* search that allows for a robot to go over another robot's path, which we call a *conflict*. We start by creating a *queue* with all the robots that move at makespan time m. While the queue is not empty, we repeat the following procedure for a robot r popped from the front of the queue. Let $q(r)$ be the number of times a robot r has been popped out of the queue. We define the *weight* of a robot r as $1 + (q(r))^2$.

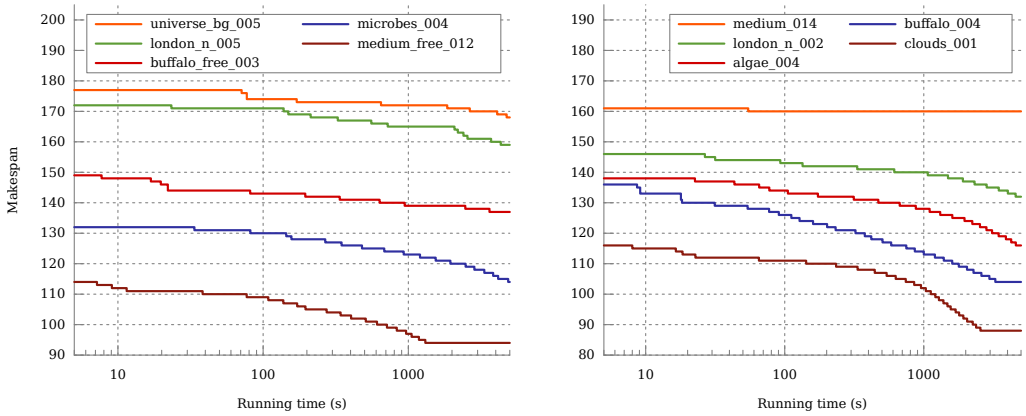


Figure 6 Improved makespan over computation time using the Conflict Optimizer.

1. Erase r's path.
2. Find a path for r from start to target that arrives no later than time $m-1$ and minimizes the weighted sum of conflicting robots.
3. Add all conflicting robots to the queue.

For sparse instances, the Conflict Optimizer can even be used to compute solutions from scratch by choosing an initial makespan and putting all the robots in the queue.

4 Results

Tables 1 and 2 show the makespan obtained using different heuristics on some selected challenge instances and the makespan lower bound. Figure 6 shows the improvement obtained by the Conflict Optimizer over a little more than one hour of execution.

The two other teams `UNIST` [10] and `gitastrophe` [6] on the podium of the CG:SHOP 2021 challenge used similar strategies, also computing an initial solution through storage networks. The methods used to optimize a solution are somewhat similar to our Feasible Optimizer, plus simulated annealing for `UNIST` and optimization of samples of k robots chosen according to their makespan, distance, or conflicts for `gitastrophe`. None of them used an optimization algorithm with a dynamic queue of robots as our Conflict Optimizer.

Table 1 Makespan of different heuristics for selected instances without obstacles.

			initial solution				optimizer		lower
instance	n	w	Gree.	Cross	Coot.	Dich.	Feas.	Conf.	bound
small_free_002	40	10	**17**	22	27	22	17	**15**	**15**
small_free_003	70	10	**20**	31	27	26	20	**16**	14
small_free_010	200	20	**34**	46	54	45	33	**32**	**32**
small_free_015	280	20	·	**60**	68	65	51	**40**	32
small_free_016	320	20	**63**	68	77	68	60	**47**	36
medium_free_007	630	30	148	**89**	103	95	81	**60**	52
medium_free_009	800	40	**93**	97	124	109	81	**71**	**71**
medium_free_012	1000	50	·	**114**	125	127	96	**94**	**94**
microbes_004	1250	50	·	**132**	159	135	125	**91**	**91**
buffalo_free_003	1440	60	·	**149**	165	158	125	**87**	78
london_night_005	1875	50	·	179	190	**173**	157	**124**	92
universe_bg_005	2000	50	·	194	198	**177**	173	**141**	82
galaxy_c2_008	3000	100	·	**198**	258	234	168	**163**	**163**
large_free_004	3938	75	·	274	276	**256**	240	**204**	127
large_free_005	5000	100	·	**260**	316	293	252	**184**	**184**
large_free_007	6000	100	·	**297**	343	325	295	**236**	189
sun_009	7500	100	·	424	395	**361**	354	**345**	187
large_free_009	9000	100	·	514	440	**391**	378	**374**	182

Table 2 Makespan of different heuristics for selected instances with obstacles.

			initial solution				optimizer		lower
instance	n	w	Gree.	Cross	Coot.	Esca.	Feas.	Conf.	bound
small_005	63	10	**27**	28	32	37	25	**20**	18
sun_000	143	20	**32**	39	46	61	29	**27**	**27**
small_011	183	20	**56**	60	70	67	48	**40**	37
small_016	276	20	·	**67**	72	79	57	**43**	36
medium_005	407	30	·	119	110	**106**	94	**74**	58
london_night_002	825	50	·	**149**	162	165	142	**94**	84
microbes_002	958	50	·	**111**	135	173	97	**89**	**89**
clouds_001	912	50	·	**117**	138	159	94	**83**	**83**
medium_014	1165	40	·	180	**161**	180	161	**151**	73
algae_004	1113	50	·	**139**	160	191	121	**84**	79
buffalo_004	1404	60	·	**136**	164	195	120	**104**	**104**
large_003	1906	100	·	**172**	224	250	**154**	**154**	**154**
large_004	2034	100	·	431	391	**381**	·	**381**	185
large_005	3223	75	·	398	**310**	317	·	**299**	141
universe_bg_007	3820	100	·	**224**	289	323	202	**184**	**184**
large_007	4706	100	·	753	497	**491**	497	**471**	215
microbes_008	5643	100	·	**329**	359	425	322	**279**	188
algae_009	7311	100	·	500	**439**	441	·	**421**	176
large_009	8595	100	·	398	**387**	566	·	**352**	176

References

1. IBM ILOG Cplex. V12. 1: User's manual for CPLEX. *International Business Machines Corporation*, 46(53):157, 2009.
2. Sándor P. Demaine, Erik D .and Fekete, Phillip Keldenich, Henk Meijer, and Christian Scheffer. Coordinated motion planning: Reconfiguring a swarm of labeled robots with bounded stretch. *SIAM Journal on Computing*, 48(6):1727–1762, 2019. doi:10.1137/18M1194341.
3. Sándor P. Fekete, Phillip Keldenich, Dominik Krupke, and Joseph S. B. Mitchell. Computing coordinated motion plans for robot swarms: The cg:shop challenge 2021, 2021. arXiv:2103.15381.
4. Ariel Felner, Roni Stern, Solomon Eyal Shimony, Eli Boyarski, Meir Goldenberg, Guni Sharon, Nathan Sturtevant, Glenn Wagner, and Pavel Surynek. Search-based optimal solvers for the multi-agent pathfinding problem: Summary and challenges. In *Tenth Annual Symposium on Combinatorial Search*, 2017.
5. Guni Sharon, Roni Stern, Ariel Felner, and Nathan R Sturtevant. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence*, 219:40–66, 2015. doi:10.1016/j.artint.2014.11.006.
6. Jack Spalding-Jamieson, Paul Liu, Brandon Zhang, and Da Wei Zheng. Coordinated motion through randomized k-opt. In *37th International Symposium on Computational Geometry, SoCG 2021*, volume 189 of *LIPIcs*, pages 64:1–64:8, 2021. doi:10.4230/LIPIcs.SoCG.2021.64.
7. Roni Stern, Nathan Sturtevant, Ariel Felner, Sven Koenig, Hang Ma, Thayne Walker, Jiaoyang Li, Dor Atzmon, Liron Cohen, TK Kumar, et al. Multi-agent pathfinding: Definitions, variants, and benchmarks. *arXiv preprint*, 2019. arXiv:1906.08291.
8. Pavel Surynek. Unifying search-based and compilation-based approaches to multi-agent path finding through satisfiability modulo theories. In *International Joint Conferences on Artificial Intelligence*, pages 1177–1183, 2019.
9. Pavel Surynek, Ariel Felner, Roni Stern, and Eli Boyarski. Efficient SAT approach to multi-agent path finding under the sum of costs objective. In *22nd European Conference on Artificial Intelligence, ECAI 2016*, volume 285 of *Frontiers in Artificial Intelligence and Applications*, pages 810–818, 2016. doi:10.3233/978-1-61499-672-9-810.
10. Hyeyun Yang and Antoine Vigneron. A simulated annealing approach to coordinated motion planning. In *37th International Symposium on Computational Geometry, SoCG 2021*, volume 189 of *LIPIcs*, pages 65:1–65:9, 2021. doi:10.4230/LIPIcs.SoCG.2021.65.

Coordinated Motion Planning Through Randomized k-Opt

Paul Liu
Department of Computer Science, Stanford University, CA, USA

Jack Spalding-Jamieson
Department of Computer Science, University of Waterloo, Canada

Brandon Zhang
Vancouver, Canada

Da Wei Zheng
Department of Computer Science, University of Illinois at Urbana-Champaign, IL, USA

Abstract

This paper examines the approach taken by team `gitastrophe` in the CG:SHOP 2021 challenge. The challenge was to find a sequence of simultaneous moves of square robots between two given configurations that minimized either total distance travelled or makespan (total time). Our winning approach has two main components: an initialization phase that finds a good initial solution, and a k-opt local search phase which optimizes this solution. This led to a first place finish in the distance category and a third place finish in the makespan category.

2012 ACM Subject Classification Computing methodologies → Motion path planning; Theory of computation → Computational geometry

Keywords and phrases motion planning, randomized local search, path finding

Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.64

Category CG Challenge

Related Version *Full Version*: `https://arxiv.org/abs/2103.15062`

Supplementary Material *Software (Source Code)*:
`https://github.com/jacketsj/cgshop2021-gitastrophe`
archived at `swh:1:dir:bbf1b4b259bd57b235c0d3774d9390e1e584ff42`

1 Introduction

For a set of unit square robots $\mathcal{R}$ each with start and target locations $\{s_r\}_{r\in\mathcal{R}}, \{d_r\}_{r\in\mathcal{R}} \subset \mathbb{Z}^2$, and a set of unit square obstacles $O \subset \mathbb{Z}^2$, the *coordinated motion planning* problem asks for an optimal sequence of simultaneous "moves" for each robot that brings the robots from their start location to their target location. At each timestep, robots can move to an adjacent grid cell or stand still (a no-op). These moves are subject to the following constraints at all timesteps:

- No robot shares a location with another robot or an obstacle.
- No robot moves into a location previously occupied by another robot, unless the other robot is moving in the same direction at that timestep.

An *optimal* sequence of moves can refer to two different criteria:

- **MAX:** The minimum **makespan**, i.e. the total number of timesteps needed.
- **SUM:** The minimum **total distance**, i.e. the total number of position-changing moves each robot takes.

We, team `gitastrophe`, explored this problem as a part of the 2021 Computational Geometry Challenge (CG:SHOP 2021). The challenge ranked teams according to each of the two different optimality criteria. Our team ranked first among all junior teams, first

according to the total distance criterion, and third according to the makespan criterion. Team Shadoks [1] ranked first according to makespan and third according to total distance, while Team UNIST [3] ranked second in both categories.

One important property of the test instances of the challenge is that feasibility is always guaranteed – no obstacles enclose the starting or target positions of any robots. In the following sections, we assume this property. We refer the reader to the survey of the challenge [2] for more details on the instances.

2 Methods

As with other teams who took part in the challenge, our strategy consisted of two phases: initialization (Section 2.1) and solution optimization (Section 2.2).

The initialization phase consisted of moving the robots out of the initial problem grid, into an intermediate state that can easily be routed to their end goals. This strategy was remarkably similar to the feasibility approaches used by both Team Shadoks [1] and Team UNIST [3]. The solution optimization phase used a novel local search strategy, which consisted of locally reordering the movements of k sampled robots while keeping all others fixed.

2.1 Initialization strategy

Our initialization strategy is simple and relies on the notion of *depth* values (see Definition 1). The depth values partition the robots into subsets which can be routed in parallel. Intuitively, once a robot is routed, it will not interfere with robots of smaller depth value.

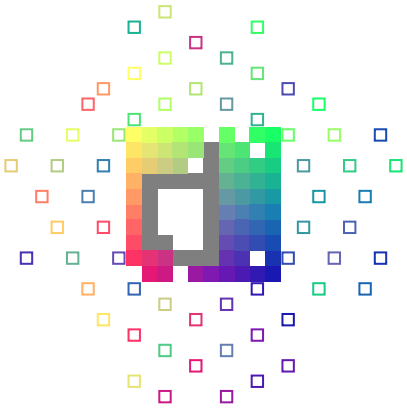
▶ **Definition 1.** *Given a set of locations $H \subset \mathbb{Z}^2$, the depth value $D(v)$ for $v \in \mathbb{Z}^2$ is computed recursively:*

- *If $v \in H$, then $D(v) = 0$.*
- *Otherwise, $D(v) = 1 + \min_u D(u)$ where the* min *is over all non-obstacle squares adjacent to v.*

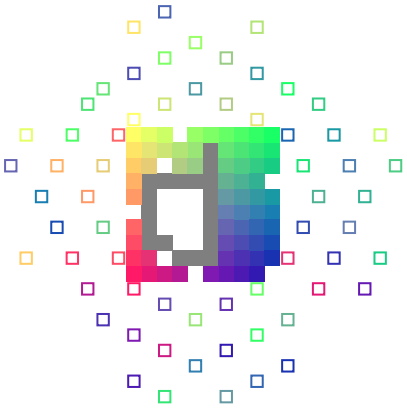
Such values are uniquely defined and can easily be found via breadth-first search.

The initialization operates in a few phases:

1. Compute a set of intermediate positions H, derived from a superset of candidate positions which we call *filler* shapes (see Figure 2). These filler shapes are located outside of the bounding box of the robots' start positions, and arranged in a way such that no intermediate positions are adjacent.
2. Compute a min-cost matching of the robots to their intermediate positions. The cost of a matching is the sum of distances from the robots' start and target positions to their matched intermediate position. To improve the initialization, we generate many more intermediate positions than robots. Figure 1 shows an example of the robots' start and target positions and the corresponding matched intermediate positions.
3. For each coordinate, compute the depth of that coordinate using H (see Definition 1). Route the robots from their start position to their corresponding intermediate positions one at a time, in order according to decreasing depth values of their start positions. The routes here are constructed to avoid the paths of the robots routed before it. In this phase we route robots with starting locations of larger depth before smaller ones.
4. Re-route the robots *from their start positions* to their target positions' in decreasing order of their depth values. We start with the paths computed in the previous phase, and reroute each robot individually. We do this by removing their path and inserting a new path respecting the current paths of all robots.



(a) `small_005` start and intermediate positions.



(b) `small_005` target and intermediate positions.

(c) `medium_005` start and intermediate positions.

(d) `medium_005` target and intermediate positions.

Figure 1 A visualization of our initializations for the instances `small_005` and `medium_005`. The grey squares are obstacles. On the left (Figures 1a and 1c), the colours of the filled boxes match the instances start positions to the intermediate positions. On the right, (Figures 1b and 1d), the colours match the target positions and intermediate positions.

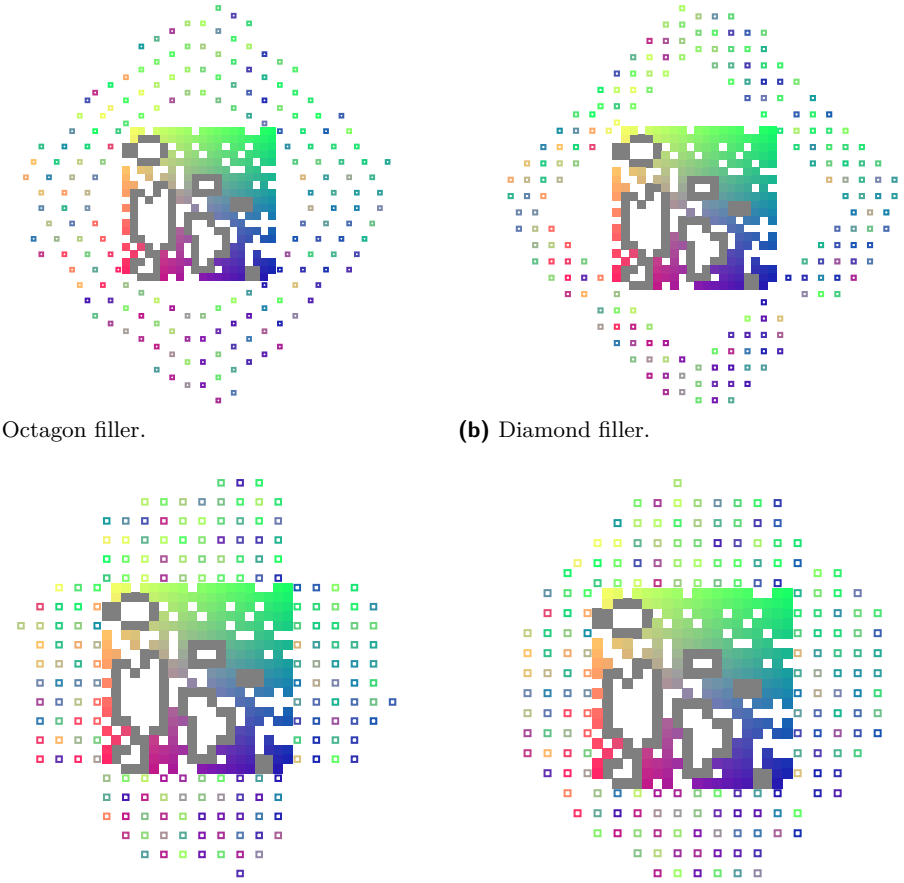
Although a natural alternative to step 4 is to simply route the robots from the intermediate locations to their target locations, it is instead much more efficient to re-route from the robots' starting locations. Given the paths produced in the third step, a feasible set of paths is guaranteed to exist in the fourth step, since it's clear that there exists a set of paths from each robots' intermediate position to their target positions: All that is needed is a path from the intermediate position to the target position that works after all other robots have finished moving. Such a path exists since robots residing at higher depth values do not block the paths of robots at lower depth values.

Initialization variations

Local search strategies are typically very sensitive to the initialization. We used a number of variations to generate a family of different initializations:

- Using different filler shapes in step 1, or different costs for matching in step 3.
- Finding random shortest paths, or approximate shortest paths for steps 3 and 4.
- Swapping the robots' start and target locations.

The optimization procedure was run on the initializations that scored the highest.



(a) Octagon filler.

(b) Diamond filler.

(c) Quad rectangle filler.

(d) Rectangle filler.

Figure 2 Visualizations of the different methods we used to generate intermediate positions for the instance `small_012`. This visualization only shows the matched potential intermediate positions, which is why the displayed intermediate positions in Figure 2d do not resemble a rectangle.

2.2 Solution optimization

As previously described, the basic paradigm of our local optimization is to choose k robots, and improve their paths while keeping the paths of all others fixed. This type of approach is often referred to as k-opt. This approach has two distinct components: (1) the choice of the k robots and (2) the improvements of their paths.

Choosing the k robots

We experimented with a few types of weighted sampling to choose the k robots:

- *Sampling by completion time.* Sample k robots without replacement, where the sampling probability for each robot is proportional to their completion time (the time at which they make their last move).
- *Sampling by closeness.* Sample the initial robot proportional to completion time, then sample $k-1$ other robots based on proximity to the initial robot's path. The proximity of two paths is the number of time steps for which the two paths are adjacent to each other.
- *Sampling by constraints.* Sample the initial robot proportional to completion time, then find a minimum completion time path from the start to the target for the sampled robot with the relaxation that this robot is allowed to move through $k-1$ other robots. The robots that the path moves through forms the $k-1$ other robots in the sample.

In our experiments, sampling by completion time and by closeness seemed to produce the best results. However, we note that team Shadoks was able to exploit a variant of sampling by constraints to win the MAX category.

Path optimization

Given a sample of k robots, one would like to jointly optimize the k robots simultaneously. However, this approach causes the state space to grow exponentially with the number of robots. Furthermore, due to the size of the grid, the state space for path finding is already quite large to begin with. Like other teams, our path finding was done in the grid-time graph where states are characterized by the positions of the robots at each time t. For the largest instance in the data set, the size of this state space is already on the order of 10^5. For these reasons, we were only able to scale to $k = 2$ in our code with the approach of joint optimization.

Instead, our main insight was to *approximate* the joint optimization by k individual single path optimizations. Our inspiration was the analogy of sorting: given the k robots, we route the robots one-by-one to their targets, where the j-th robot routed ($j \leq k$) respects the path of robots $1 \dots j-1$. Crucially, this avoids the exponential blow-up of the state space as we're routing each robot from start to target in succession. The problem then reduces to finding a good ordering of the k robots to reroute. We had the most success by simply ordering the robots by decreasing completion time. If an ordering was infeasible (e.g. due to some subset of the k robots completely blocking off the path from the remaining robots) or not an improvement, it was discarded.

During the competition we found that there were advantages to relatively smaller values of k for faster computation. For our approximate joint optimization, larger k helped to get out of bad local optima. We iteratively used all values of k between 1 and 7 during the challenge.

Since the main component of our algorithm is path-finding, we used a number of practical techniques to reduce its cost. We use A* with a bucket priority queue, where the heuristic function was taken to be either the Manhattan distance or the shortest path distance in the graph with only obstacles without robots. To further speed up the search when we have an initial feasible solution, we limit the path-finding algorithm to search locally around the robot's original path, by enforcing that no robot may deviate more than R steps away from the set of positions forming their initial path for a fixed parameter R. Since the search is centred around the initial solution, the solution space is guaranteed to have a feasible solution for any value of R.

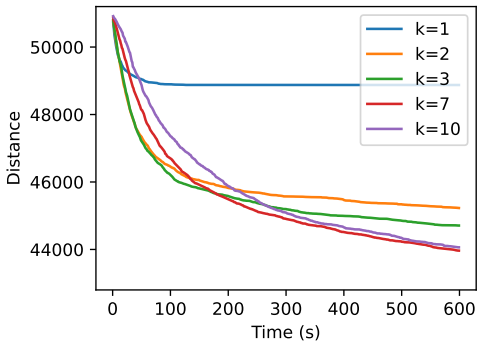
The optimizations above are agnostic to the optimization objective and allowed for large gains following the initialization phase for both SUM and MAX.

3 Results

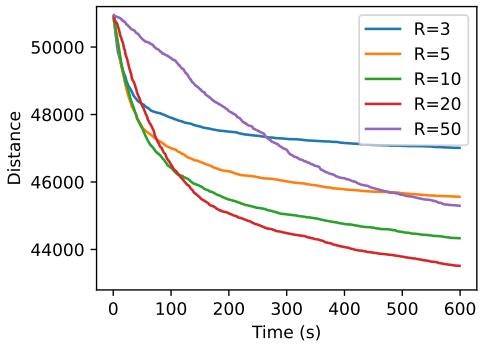
3.1 Computational environment

For the most part, our experiments were done on a desktop with a Core i7-2600. In the last three weeks of the competition we used Stanford's Sherlock High-Performance Computing Cluster for SUM optimization. The University of Waterloo's Multicore lab also generously donated some night-time compute during the last week of the competition in the form of two EPYC 7662s.

3.2 Experimental results

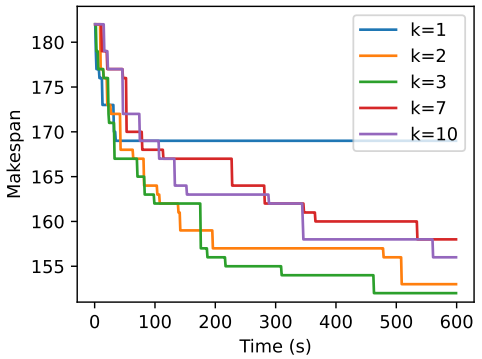


(a) SUM optimization varying k where $R = 20$.

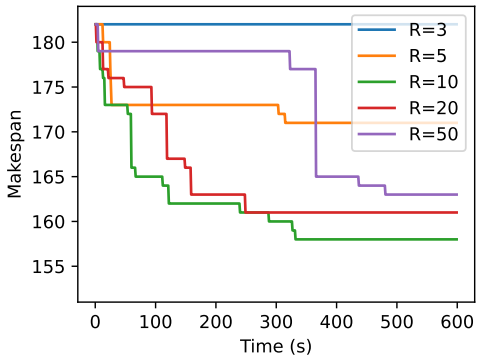


(b) SUM optimization varying R where $k = 7$.

Figure 3 Varying k and R for SUM optimization on `microbes_004`. Final SUM: 43437.



(a) MAX optimization varying k where $R = 20$.



(b) MAX optimization varying R where $k = 7$.

Figure 4 Varying k and R for MAX optimization on `microbes_004`. Final MAX: 126.

To empirically determine the effect of k and R had on the optimization, we ran our algorithm on the instance `microbes_00004`. As seen in Figure 3a, different values of k had an effect on the rate of convergence and the value of the local minimum found by our strategy. Experimentally, values of k greater than 10 did not improve our scores and also ran slower. For MAX (in Figure 4a), improvements were much more discrete.

As we varied R, we had a similar trade-off between performance and runtime for each optimization step of our algorithm (Figures 3b and 4b). Empirically, choosing R to be around 20 achieved the best balance for most of the instances.

Figure 5 compares the solutions we computed during the competition to the trivial lower bound, which is given by the maximum shortest-path distance for SUM and the sum of all shortest-path distances for MAX. There is a clear trend that score decreases as density increases (and as n increases, to a lesser extent). This could be due to two factors: first, that the trivial lower bound becomes a worse approximation of the optimal score as these values increase; second, that our algorithm performs poorer on these instances, because more computational time is required and local changes to k robots at a time are insufficient to traverse the solution space.

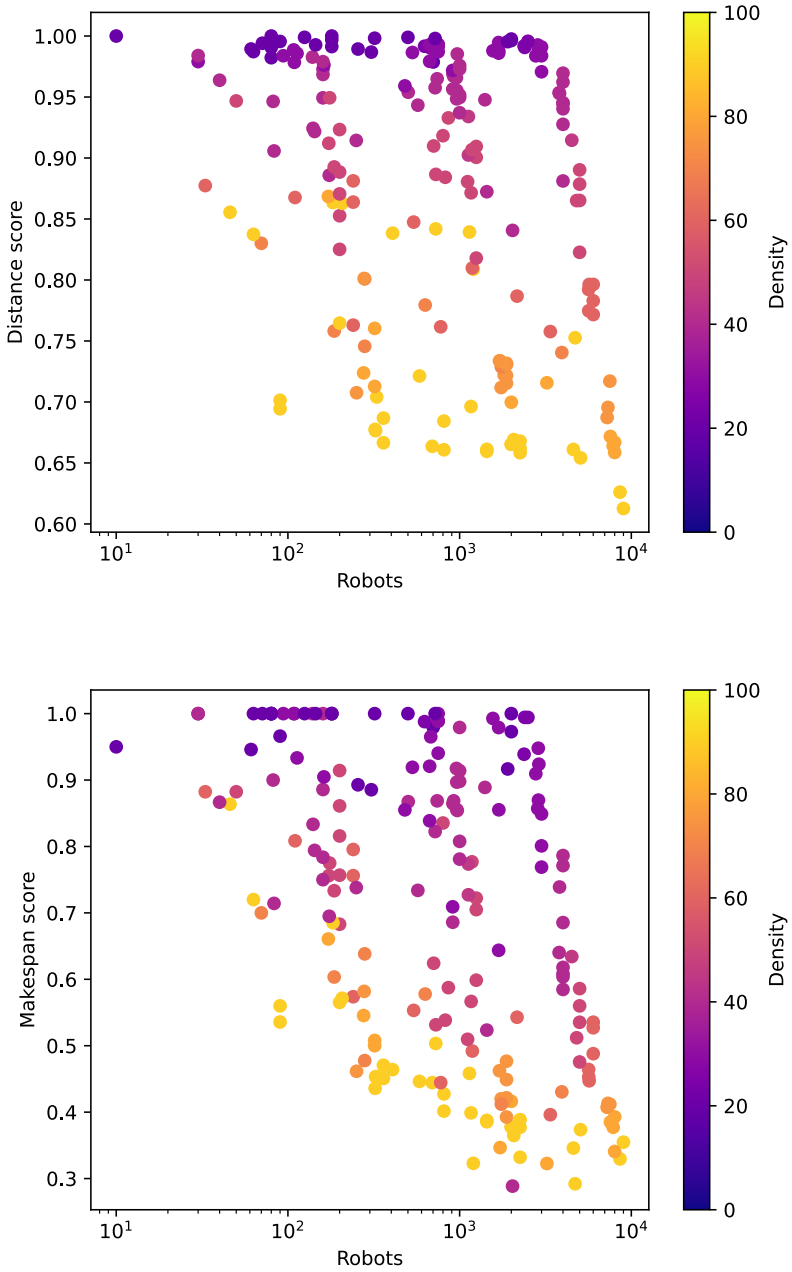


Figure 5 Plots of $\frac{\text{score}}{\text{lower bound}}$ versus n for both SUM and MAX. Each point corresponds to one instance. The density of the instance, as defined by the contest organizers, is indicated by colour.

References

1 Loïc Crombez, Guilherme D. da Fonseca, Yan Gerard, Aldo Gonzalez-Lorenzo, Pascal Lafourcade, and Luc Libralesso. Shadoks Approach to Low-Makespan Coordinated Motion Planning. In *37th International Symposium on Computational Geometry (SoCG 2021)*, volume 189 of *LIPIcs*, pages 63:1–63:9, 2021. doi:10.4230/LIPIcs.SoCG.2021.63.

2 Sándor Fekete, Phillip Keldenich, Dominik Krupke, and Joseph S. B. Mitchell. Computing Coordinated Motion Plans for Robot Swarms: The CG:SHOP Challenge 2021. *CoRR*, abs/2103.15381, 2021. arXiv:2103.15381.

3 Hyeyun Yang and Antoine Vigneron. A Simulated Annealing Approach to Coordinated Motion Planning. In *37th International Symposium on Computational Geometry (SoCG 2021)*, volume 189 of *LIPIcs*, pages 65:1–65:9, 2021. doi:10.4230/LIPIcs.SoCG.2021.65.

A Simulated Annealing Approach to Coordinated Motion Planning

Hyeyun Yang
Ulsan National Institute of Science and Technology, South Korea

Antoine Vigneron[1]
Ulsan National Institute of Science and Technology, South Korea

Abstract

The third computational geometry challenge was on a coordinated motion planning problem in which a collection of square robots need to move on the integer grid, from their given starting points to their target points, and without collision between robots, or between robots and a set of input obstacles. We designed and implemented an algorithm for this problem, which consists of three parts. First, we computed a feasible solution by placing middle-points outside of the minimum bounding box of the input positions of the robots and the obstacles, and moving each robot from its starting point to its target point through a middle-point. Second, we applied a simple local search approach where we repeatedly delete and insert again a random robot through an optimal path. It improves the quality of the solution, as the robots no longer need to go through the middle-points. Finally, we used simulated annealing to further improve this feasible solution. We used two different types of moves: We either tightened the whole trajectory of a robot, or we stretched it between two points by making the robot move through a third intermediate point generated at random.

2012 ACM Subject Classification Theory of computation → Computational geometry; Computing methodologies → Motion path planning

Keywords and phrases Path planning, simulated annealing, local search

Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.65

Category CG Challenge

Funding This work was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2017R1D1A1B04036529).

1 Introduction

In this paper, we present our solution to the third computational geometry challenge, which was on a coordinated motion planning problem [2, 3]. We first briefly describe this problem. A set of N robots, modeled as unit squares, need to move on the integer grid $\mathbb{Z}^2$, from their starting positions $s_1, \ldots, s_N \in \mathbb{Z}^2$ to their target positions $d_1, \ldots, d_N \in \mathbb{Z}^2$. A (possibly empty) set $\mathcal{O} \subset \mathbb{Z}^2$ of obstacles is also given. We denote by $p_i(t) = (x_i(t), y_i(t)) \in \mathbb{Z}^2$ the position of robot i at time $t \in \mathbb{N}$. At each time $t \in \mathbb{N}$, the robot may either stay at the same position, or move to one of the four neighboring squares, hence we have $p_i(t+1) - p_i(t) \in \{(0,0), (-1,0), (0,-1), (0,1), (1,0)\}$.

While moving, each robot must avoid collision with obstacles or other robots. In particular, for any robot i and any time $t \in \mathbb{N}$, we must have $p_i(t) \notin \mathcal{O}$ and $p_i(t) \neq p_j(t)$ for all $j \neq i$. In addition, a constraint is enforced in order to model coordinated movement: a robot i can only move to the position previously occupied by another robot j if they move in the same direction. More precisely, if $p_i(t+1) = p_j(t)$, then we must have $p_i(t+1) - p_i(t) = p_j(t+1) - p_j(t)$.

[1] Corresponding author.

The *length* of the trajectory of robot i is the number of times robot i moves, in other words it is $|\{t \in \mathbb{N} \mid p_i(t+1) \neq p_i(t)\}|$. Its *completion time* $t_C(i)$ is the time when robot i ceases to move; in other words, it is the minimum time $t_C(i)$ such that for all $t \geq t_C(i)$, we have $p_i(t) = d_i$. The *makespan* is the maximum completion time.

The solutions to this coordinated motion planning problem were scored according to two criteria: we should either minimize the makespan (MAX), or minimize the sum of the lengths of the paths (SUM). Our team (UNIST) ranked second according to both criteria, out of 17 teams participating in the contest. Team Shadoks [1] ranked first according to MAX and third according to SUM, while team gitastrophe [4] ranked first according to SUM and third according to MAX. More information on the contest can be found in the survey by Fekete et al. [3]

2 Data structure

Our data structure consists of two 3-dimensional arrays G and H. The array G is a 3D-array of 8-bit integers, where $G[x, y, t] = -1$ if (x, y) is an obstacle, $G[x, y, t] = 0$ if the cell (x, y) is empty at time t, and if $(x, y) = p_i(t)$ for some i, then $G[x, y, t]$ records $p_i(t + 1) - p_i(t)$. In other words, if robot i is located at (x, y) at time t, then $G[x, y, t]$ records the direction taken by robot i. We encode this direction as an integer in $\{1, \ldots, 5\}$, where 5 means that robot i does not move. (In particular, G and H do not record robot numbers.) The array H is a 3D-array of 16-bit integers, which is only needed in the SUM version.

We chose this data structure in order to minimize memory usage, as we feared that large instances would not fit in RAM otherwise. It turns out that we only used a small percentage of our RAM even for the largest instances (less than 7%), so we could have afforded a larger data structure.

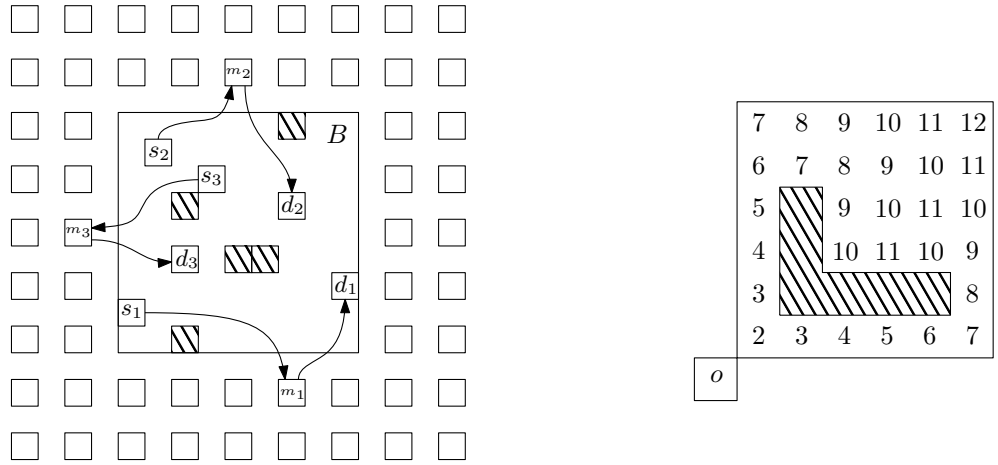
For the sake of analyzing our algorithms, we use w to denote the size of G in either dimension. The reason is that, for the problem instances given in the contest, the makespan of the solutions we constructed was not much larger than the length and width of the grid we used – less than a factor 10. So we may assume that G is a $w \times w \times w$-array. Our data structure allows us to do the following.

Deletion. We can delete the trajectory of robot i from G in $O(w)$ time. It suffices to follow the direction given by $G[x, y, t]$ from the starting point s_i.

Insertion. Assuming robot i is not yet recorded in G, and there is a feasible path from s_i to d_i, we can insert in $O(w^3)$ time a feasible trajectory of robot i that either minimizes the completion time $C_p(i)$, or minimizes its length. It can be done by a simple sweep of G by increasing values of t, and storing in $G[x, y, t]$ the direction of the move from the parent (if any) of (x, y, t) as an integer in $\{11, \ldots, 15\}$. In the SUM version, we also record in $H[x, y, t]$ the length of the shortest feasible trajectory to (x, y, t). At the end of the sweep, we reconstruct the path backwards, and we remove all values larger than 10 from G.

3 Computing a feasible solution

In order to compute a feasible solution, we construct a middle-point m_i for each robot i, such that there is a feasible path from s_i to d_i through m_i. (See Figure 1a.) Let B be the minimum bounding box of the starting points, target points and obstacles. The middle-points are chosen from the set of grid points outside B, not adjacent to B, and whose x and y-coordinates are even.



(a) (b)

Figure 1 (a) Robot i goes from s_i to d_i through m_i. (b) The geodesic distance from o.

In a first stage, we move all the robots to their middle-points, and in a second stage we move them from their middle-points to their target points. In order to do this, we insert the robots one by one in our 3D array G, by applying the insertion procedure from Section 2.

One difficulty here is that a robot can be blocked by other robots that are still at their starting position, so we need to move the robots in an appropriate order. To this end, we compute the geodesic distances from an arbitrary point o outside B. (See Figure 1b.) We sort the robots by *increasing* geodesic distance, and insert them in this order. This ensures that there is always a feasible path from s_i to m_i for each i. In the second stage, we proceed in the same way, except that we proceed by *decreasing* geodesic distance from o to the target points. This approach finds a feasible solution to all the input instances from the contest, as all robots are in the unbounded face of $\mathbb{Z}^2 \setminus \mathcal{O}$.

For each robot i, we have several choices for the middle-point m_i. We tried a few possibilities. The one that most often gives the best results was to choose the available middle-point that minimizes the sum of the Manhattan distances to s_i and d_i.

As we run the deletion and insertion procedures $2w$ times, we obtain a feasible solution in $O(w^4)$ time. We were able to compute a feasible solution for each instance in less than 2 hours.

4 Simple local search

In order to improve the feasible solution from Section 3, we can simply pick a robot i, delete it from G and insert it again using the procedures from Section 2. It may give a quite large improvement for this robot, as it no longer has to go through its middle-point m_i. We call this operation a *tightening* move. (See Figure 2.)

A first approach is to repeatedly tighten the path of a robot chosen at random. We implemented it in a slightly different way, which produced better results. We first compute a random permutation of the robots. Then we go through this random list, and perform a tightening operation on the current robot. If, at the end of the list, no improvement was made, we return the current solution. Otherwise, we compute a new random permutation and repeat the process with the new permutation. The procedure above can be restarted from the original feasible solution several times, as it does not always produce the same solution.

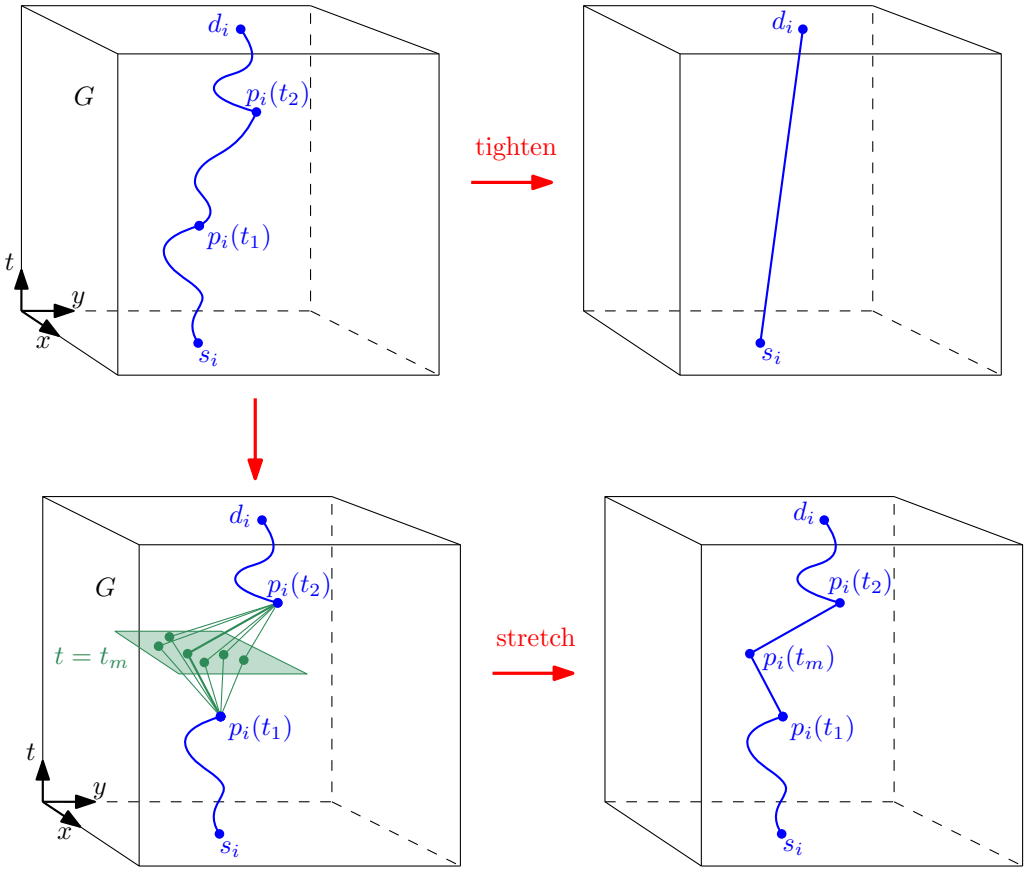


Figure 2 Tightening and stretching moves.

5 Simulated annealing

One drawback of the local search approach described in Section 4 is that the only type of moves that is allowed consists in shortening the whole trajectory of one robot, and thus it can easily be trapped in a local minimum. To remedy this, we used a simulated annealing approach [5] that uses two types of moves: tightening moves as described in Section 4, and *stretching* moves, which can make a trajectory longer.

We now describe stretching moves. (See Figure 2.) As mentioned above, we assume that G is a $w \times w \times w$ array. Let i be a robot, and let t_1, t_2 be integers such that $0 \leq t_1 \leq t_2 < n$. We will see later how we generate t_1 and t_2 at random. We first delete the trajectory of i between $p_i(t_1)$ and $p_i(t_2)$. Let t_m be chosen uniformly at random between t_1 and t_2. We compute all of the cells (x, y, t_m) of G that are reachable from $p_i(t_1)$, and that are reachable backwards from $p_i(t_2)$. We pick one of these points uniformly at random, which we denote by q. Then we connect q to $p_i(t_1)$ and $p_i(t_2)$ through shortest paths, computed in the same way as we did in the insertion operation from Section 2.

Let $\delta = t_2 - t_1$. As we only need to sweep a sub-array of size at most $2\delta \times 2\delta \times \delta$, and we need linear time to find $p_i(t_1)$ and $p_i(t_2)$, the stretching operation takes time $O(w + \delta^3)$.

We now explain how we generate t_1 and t_2. We first set $\delta = \min(w - 1, \lfloor \alpha\sqrt{1/x} \rfloor)$, where α is a constant larger than 2, and x is a random floating point number in $(0, 1]$. Then we generate t_1 as a random number chosen uniformly between 0 and $n - 1 - \delta$, and we set $t_2 = t_1 + \delta$. This approach ensures that, for $k < n - 1$, we have $\Pr(\delta = k) = \Theta(1/k^3)$, and thus $E[\delta^3] = \Theta(w)$. It follows that our stretching operation takes $O(w)$ expected time.

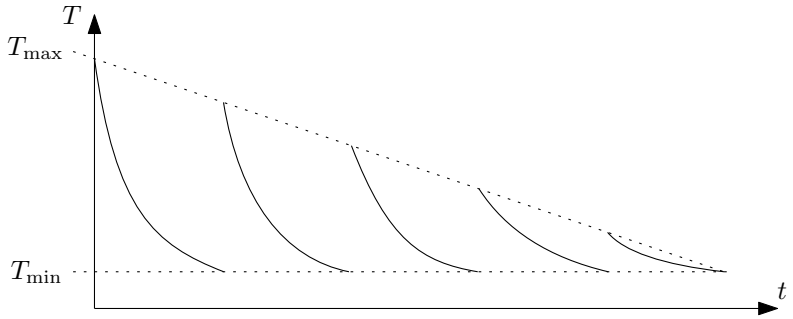


Figure 3 A cooling schedule with $n_{\text{cycle}} = 5$.

We use as the objective function a score that is either the makespan, or the sum of the lengths of the paths divided by the number of robots. A move that improves the score is always accepted by the algorithm, but a move that increases it is accepted with probability $\exp(-\Delta/T)$, where T is the current temperature and Δ is the score increase. We used a cooling schedule consisting of n_{cycle} cycles of n_{iter} iterations each, such that the temperature decreases exponentially within each cycle, and the maximum temperature decreases linearly. (See Figure 3.) We did not try other cooling schedules, but we tried different number of cycles and iterations per cycle.

Choice of parameters. We determined the values of the various parameters of the algorithm by trying it on several instances, varying the parameters and comparing the results.

We found that $\alpha = 5$ did better than other values on most instances. Regarding the choice of the moves, we generated a tightening move or a stretching move with the same probability 1/2, which implies that the algorithm is spending much more time on tightening. For the SUM version, in some cases, we obtained better results by generating a tightening move with probability 1/1 000, which means that the algorithm spends roughly the same amount of time on stretching and tightening.

We chose the minimum temperature to be $T_{\min} = 0.0001$. The maximum $T_{\max}$ was between 0.03 and 1 for the MAX version, and between 0.001 and 0.02 for the SUM version. The number of cycles was between 500 and 5 000. The number of iterations per cycle (typically 10 000-1 000 000) was adjusted depending on the time we wanted the computation to run for.

We tried different sets of moves; for instance, we tried to do tightening moves in a time window $[t_1, t_2]$ instead of $[0, n]$. We also tried to use a smoothed objective function, using a linear combination of the makespan and the sum of the lengths. Unfortunately, it did not seem to help.

6 Experimental results

We implemented the algorithms above in C++ and ran them on a server (4 Intel Xeon E5-2695 V4, 2.1 GHz, 72 cores in total). We denote by F, LS and SA the algorithm that generates the initial feasible solution (Section 3), the simple local search algorithm (Section 4) and the simulated annealing algorithm (Section 5), respectively.

We computed a feasible solution for each instance using F, which has two versions: one for SUM and one for MAX. Then we ran LS on each of these feasible solutions. After this, we ran SA on the solution given by LS, which is the approach denoted by LS+SA. Alternatively, we ran SA directly from the solution given by F.

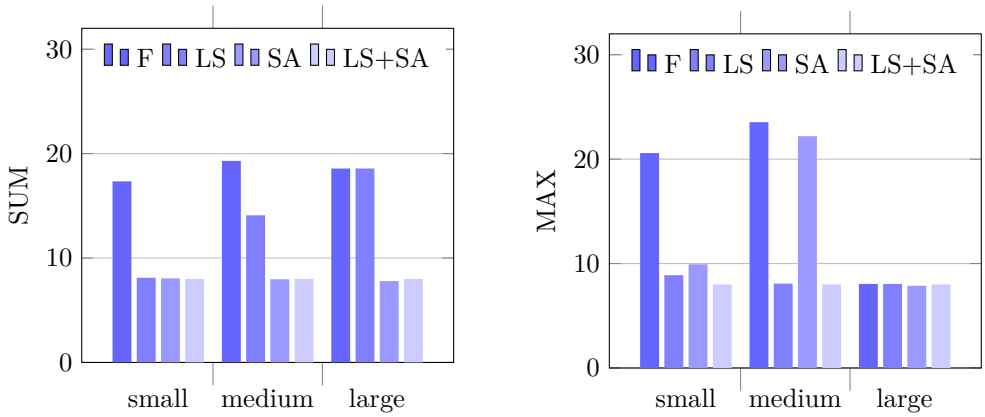


Figure 4 Performance of our algorithms on 8 small, 8 medium and 8 large-size instances. Results are normalized according to the score for LS+SA.

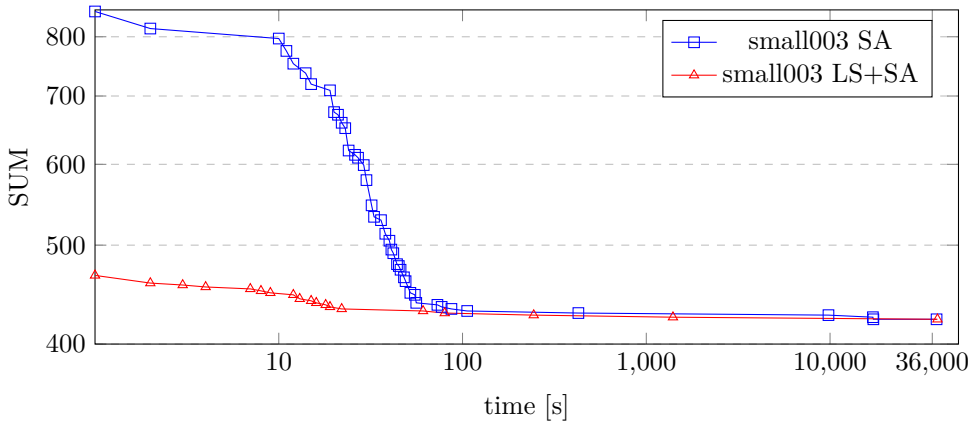


Figure 5 Best sum of lengths (SUM) until time t for the instance small_003_10x10_90_46.

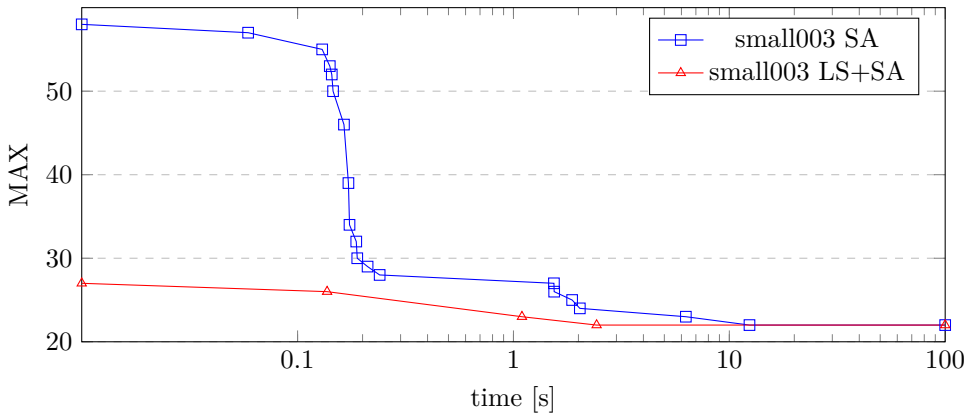


Figure 6 Best makespan (MAX) until time t for the instance small_003_10x10_90_46.

Figure 4 shows the results on 24 instances grouped into small, medium and large size, after running either SA for 8 days, or LS+SA for a total of 8 days (4 days LS, 4 days SA). Table 1 and Table 2 record the related data. Figure 5 and Figure 6 show the score improvement over time for one instance.

These results suggest that we could run SA directly after F, instead of running LS in between. One point of using LS during the contest was that it does not use any parameter, so we could run it until completion, or for large instances, stop it after a few day. Figure 4 shows that SA only does clearly worse than LS+SA for the MAX version of medium-size instances, which suggest that we may not yet have found the best parameters in this case.

Table 1 A few instances with the scores obtained after applying our algorithms (SUM).

instance name	nb. of robots	normalized score (SUM)				score (SUM)
		F	LS	SA	LS+SA	LS+SA
small instances						
medium_free_004	480	2.86	1.03	1.02	1	11300
redblue_00002	669	1.99	1.01	1	1	25083
galaxy_c_00005	750	1.97	1.01	1	1	27107
galaxy_c2_00005	860	1.96	1.01	1	1	37413
microbes_00002	958	2.22	1.01	1	1	34754
large_001	1563	2.04	1	1	1	79793
medium_018	1993	2.32	1	1	1	98984
microbes_00006	2500	1.95	1	1	1	169912
sum		17.31	8.07	8.02	8	
medium-size instances						
universe_bg_00006	3000	2.08	1	1	1	217456
sun_00006	3796	2.38	1	0.99	1	268569
algae_00007	4000	2.33	1	1.00	1	275157
redblue_00009	4500	2.40	1	0.99	1	322346
galaxy_c_00008	5000	2.54	2.54	0.99	1	356037
london_night_00008	5648	2.44	2.44	0.98	1	444974
london_night_00009	6000	2.58	2.58	0.98	1	462928
microbes_00009	6000	2.50	2.50	0.99	1	468081
sum		19.25	14.06	7.92	8	
large instances						
clouds_00009	7229	2.33	2.33	0.97	1	641904
algae_00009	7311	2.39	2.39	0.97	1	648026
sun_00009	7500	2.42	2.42	0.97	1	675357
galaxy_c2_00009	7555	2.32	2.32	0.97	1	694166
galaxy_c_00009	7838	2.31	2.31	0.97	1	740212
universe_bg_00009	8000	2.29	2.29	0.97	1	755800
large_009	8595	2.22	2.22	0.96	1	876814
large_free_009	9000	2.25	2.25	0.96	1	911623
sum		18.53	18.53	7.74	8	

Table 2 A few instances with the scores obtained after applying our algorithms (MAX).

instance name	nb. of robots	normalized score (MAX)				score (MAX)
		F	LS	SA	LS+SA	LS+SA
small instances						
medium_free_004	480	3	1.2	0.98	1	60
redblue_00002	669	2.35	1.05	0.95	1	91
galaxy_c_00005	750	2.22	1.1	1.09	1	87
galaxy_c2_00005	860	2.67	1.19	1	1	105
microbes_00002	958	3.01	1.11	1.02	1	90
large_001	1563	2.29	1.01	1.94	1	137
medium_018	1993	2.87	1.2	0.98	1	190
microbes_00006	2500	2.14	1	1.94	1	179
sum		20.58	8.88	9.93	8	
medium-size instances						
universe_bg_00006	3000	2.25	1.02	2.17	1	198
sun_00006	3796	2.69	1	2.54	1	214
algae_00007	4000	2.92	1	2.78	1	206
redblue_00009	4500	2.74	1.02	2.62	1	221
galaxy_c_00008	5000	2.62	1	2.55	1	265
london_night_00008	5648	3.47	1.01	2.88	1	318
london_night_00009	6000	3.52	1	3.45	1	292
microbes_00009	6000	3.3	1.01	3.17	1	287
sum		23.54	8.08	22.19	8	
large instances						
clouds_00009	7229	1	1	0.98	1	1099
algae_00009	7311	1	1	0.98	1	1195
sun_00009	7500	1	1	0.94	1	1094
galaxy_c2_00009	7555	1.01	1.01	0.99	1	1095
galaxy_c_00009	7838	1	1	0.98	1	1261
universe_bg_00009	8000	1	1	0.99	1	1132
large_009	8595	1	1	0.99	1	1348
large_free_009	9000	1	1	0.98	1	1334
sum		8.05	8.05	7.87	8	

References

1 Loïc Crombez, Guilherme D. da Fonseca, Yan Gerard, Aldo Gonzalez-Lorenzo, Pascal Lafourcade, and Luc Libralesso. Shadoks approach to low-makespan coordinated motion planning. In *37th International Symposium on Computational Geometry (SoCG 2021)*, volume 189 of *LIPIcs*, pages 61:1–61:9, 2021. doi:10.4230/LIPIcs.SoCG.2021.61.

2 Erik D. Demaine, Sándor P. Fekete, Phillip Keldenich, Henk Meijer, and Christian Scheffer. Coordinated motion planning: Reconfiguring a swarm of labeled robots with bounded stretch. *SIAM J. Comput.*, 48(6):1727–1762, 2019. doi:10.1137/18M1194341.

3 Sándor P. Fekete, Phillip Keldenich, Dominik Krupke, and Joseph S. B. Mitchell. Computing coordinated motion plans for robot swarms: The cg:shop challenge 2021, 2021. arXiv:2103.15381.

4 Jack Spalding-Jamieson, Paul Liu, Brandon Zhang, and Da Wei Zheng. Coordinated motion planning through randomized k-opt. In *proc. 37th International Symposium on Computational Geometry*, volume 189 of *LIPIcs*, pages 64:1–64:8, 2021. doi:10.4230/LIPIcs.SoCG.2021.64.

5 Peter van Laarhoven and Emile Aarts. *Simulated Annealing: Theory and Applications*. Springer Netherlands, 1987.