Report from Dagstuhl Seminar 23211

Scalable Data Structures

Gerth Stølting Brodal[*1], **John Iacono**[*2], **László Kozma**[*3], **Vijaya Ramachandran**[*4], **and Justin Dallant**[†5]

1 **Aarhus University, DK.** gerth@cs.au.dk
2 **UL – Brussels, BE.** john.iacono@ulb.be
3 **FU Berlin, DE.** laszlo.kozma@fu-berlin.de
4 **University of Texas – Austin, US.** vlr@cs.utexas.edu
5 **UL – Brussels, BE.** Justin.Dallant@ulb.be

Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 23211 "Scalable Data Structures". Data structures enable the organization, storage and retrieval of data across a variety of applications. As they are deployed at unprecedented scales, data structures can profoundly affect the efficiency of almost all computational tasks. The study of data structures thus continues to be an important and active area of research with an interplay between data structure design and analysis, developments in computer hardware, and the needs of modern applications. The extended abstracts included in this report give a snapshot of the current state of research on scalable data structures and establish directions for future developments in the field.

Seminar May 21–26, 2023 – https://www.dagstuhl.de/23211
2012 ACM Subject Classification Theory of computation → Data structures design and analysis; Theory of computation → Design and analysis of algorithms; Theory of computation → Parallel algorithms
Keywords and phrases algorithms, big data, computational models, data structures, GPU computing, parallel computation
Digital Object Identifier 10.4230/DagRep.13.5.114

1 Executive summary

Gerth Stølting Brodal (Aarhus University, DK)
John Iacono (UL – Brussels, BE)
László Kozma (FU Berlin, DE)
Vijaya Ramachandran (University of Texas – Austin, US)

About the seminar

The scale at which data is generated and processed is increasing unabated and novel applications continue to arise, posing new challenges for data structure design and analysis. The performance of data structures can dramatically affect the overall efficiency of computing systems, motivating research on scalable data structures along the entire spectrum from purely theoretical to purely empirical.

[*] Editor / Organizer
[†] Editorial Assistant / Collector

The focus of data structure research has continuously shifted to better align with the changing realities of the underlying hardware (e.g. by refining computational models to capture memory hierarchies and parallelism), and the requirements of applications (e.g. by finding the right input models, novel modes of operation, and special requirements such as data privacy or adapting to side-information). Suitable data structuring abstractions have often been crucial components of algorithmic breakthroughs, for instance in static or dynamic graph algorithms, e.g. for maximum flow or minimum spanning trees. Research on classical problems and long-standing open questions continues, with surprising recent improvements, e.g. for list labeling.

Data structure research has been a core part of computer science from the beginnings, and the field appears as vibrant as ever, with research continuing on deep old questions, as well as on new directions reflecting the changing realities of the computational landscape.

This seminar was the 15th in a series of loosely related Dagstuhl Seminars on data structures, bringing together researchers from several research directions to illuminate various aspects and solutions to the problem of data structure scalability. Following the previous, purely virtual meeting, the seminar was organized as a fully on-site event.

Topics

The presentations covered both advances in classic data structure fields, as well as insights that addressed the scalability of computing in different models of computation and across a diverse range of applications.

Parallelism was an important theme of the seminar. Blelloch (Section 4.1) discussed possible ways of making parallelism a core part of a computer science curriculum, Agrawal (Section 4.12) talked about incorporating data structures in parallel algorithms, and Sun (Section 4.5) presented algorithms for Longest Increasing Subsequence, building on parallel data structures.

Classic questions of data structure design and analysis in the comparison- or pointer-based models were the topic of multiple talks. Tarjan (Section 4.17) discussed various self-adjusting heaps and new results on their amortized efficiency. Munro (Section 4.24) talked about the ordered majority problem. Sorting was the subject of a series of talks: Wild (Section 4.3) talked about new, practically efficient sorting algorithms used in Python. Nebel (Section 4.6) talked about the practical efficiency of Lomuto's QuickSort variant. Pettie (Section 4.21) studied efficiently sorting inputs with pattern-avoiding properties, and Jacob (Section 4.27) studied variants of the sorting problem with priced comparisons.

Several talks reflected the central importance of hashing in scalable data structures, giving a broad picture of modern developments in this area. Conway (Section 4.2) presented Iceberg Hashing and Mosaic Pages. Sanders (Section 4.7) presented Sliding Block Hashing. Farach-Colton (Section 4.9) considered a simplified hash table design with strong guarantees. Bercea (Section 4.18) presented a data structure for incremental stable perfect hashing, Johnson (Section 4.22) presented Maplets, and Even (Section 4.29) talked about dynamic filters with one memory access.

The very recent $O(\log^{3/2} n)$-time online list labeling result was presented by Wein (Section 4.28) and results for the online list labeling problem with machine learning advice were presented by Singh (Section 4.8).

Static and dynamic graph algorithms were the topic of several presentations. Kyng (Section 4.12) talked about dynamic spanners and data structuring problems arising in the context of minimum cost flow. King (Section 4.13) presented algorithms for dynamic connectivity and Rotenberg (Section 4.15) presented dynamic graph algorithms that are adaptive to sparsity of the input. For data structures in string problems, Gørtz (Section 4.20) discussed regular expression matching and Kopelowitz (Section 4.16) presented speedups of the dynamic program for the Dyck edit distance problem.

Multiple talks reflected the interplay between data structures and computational geometry: Dallant (Section 4.30) spoke about conditional lower bounds for dynamic geometric problems, Oh (Section 4.25) presented an algorithm for the planar disjoint paths problem, and Arseneva (Section 4.23) talked about morphing graph drawings, including results obtained jointly with Oh during the seminar.

Possible computational models for designing algorithms for GPUs were addressed by Sitchinava (Section 4.19) and models of in-memory processing were presented by Silvestri (Section 4.26).

Phillips (Section 4.10) presented the design and analysis of a large-scale stream monitoring system. Liu (Section 4.4) discussed the role of scalable data structures in the context of differential privacy for graph data. Xu (Section 4.11) presented a search-optimized layout for packed memory arrays.

Final Thoughts

The organizers would like to thank the Dagstuhl team for their continuous support and also thank all participants for their contributions to this seminar. Following the earlier virtual seminar, the current (15th) seminar was fully on-site. The opportunity to personally meet and interact was highly appreciated by the community, as reflected by a very strong response to the first round of invitations and subsequent positive feedback. The seminar fills a unique need in bringing together data-structures-researchers from around the world and facilitating collaboration and exchange of ideas between them.

Earlier seminars in the series had few female participants. An important focus of the previous and the current seminar was to significantly increase female attendance. In the current seminar, 48% of the invited participants were female, resulting in a 37% female attendance. Another important focus of the seminar is to encourage the interaction between senior and early career researchers, the latter comprising 27% of the invited participants and 32% of the eventual attendees.

In the post-seminar survey the diversity of junior/senior and female/male participants were both appreciated, respondents also drawing attention to the importance of tactful and clear communication on these matters. The survey respondents also praised the coverage of a diverse range of research topics, as well as the mix between theoreticians and more applied researchers.

2 Table of Contents

3 Seminar program

Sunday May 21, 2023	
18:00	*Dinner buffet*
Monday May 22, 2023	
07:30	*Breakfast*
09:00	*Opening & Introductions*
10:30	*Coffee break*
11:00	*Should We Teach Parallelism throughout Undergraduate Algorithm Courses?* Guy E. Blelloch
12:15	*Lunch*
15:30	*Coffee & Cake*
16:00	*Iceberg Hashing and Mosaic Pages: A Data-Structural Approach to Faster Virtual Address Translation* Alexander Conway
16:35	*Quicksort, Timsort, Powersort – Python's new Sorting Algorithm* Sebastian Wild
17:10	*Scalable Data Structures for Privacy on Graphs* Quanquan C. Liu
18:00	*Dinner*
Tuesday May 23, 2023	
07:30	*Breakfast*
09:00	*Parallel Longest Increasing Subsequence and van Emde Boas Trees* Yihan Sun
09:30	*Lomuto's Comeback or the Unpredictability of Program Efficiency* Markus E. Nebel
10:00	*Sliding Block Hashing* Peter Sanders
10:30	*Coffee break*
11:00	*Open problem session*
12:15	*Lunch*
15:30	*Coffee & Cake*
16:00	*Online List Labeling with Predictions* Shikha Singh
16:24	*Simple Hash Tables* Martin Farach-Colton
16:48	*Write-Optimized Algorithms for Stream Monitoring* Cynthia A. Phillips
17:12	*Optimizing Search Layouts in Packed Memory Arrays* Helen Xu
17:38	*Using Data Structures within Parallel Algorithms* Kunal Agrawal
18:00	*Dinner*

Wednesday May 24, 2023

07:30	*Breakfast*
09:00	*Dynamic Spanners* Rasmus Kyng
09:30	*Batch Parallel fast Worst Case Dynamic Connectivity* Valerie King
10:00	*Sparsity-Adaptive Dynamic Graph Algorithms* Eva Rotenberg
10:30	*Coffee break*
11:00	*The k-Dyck Edit Distance Problem* Tsvi Kopelowitz
11:30	*Heaps* Robert Tarjan
12:00	*Group picture*
12:15	*Lunch*
14:00	*Hike*
15:30	*Coffee & Cake*
18:00	*Dinner*

Thursday May 25, 2023

07:30	*Breakfast*
09:00	*An Extendable Data Structure for Incremental Stable Perfect Hashing* Ioana-Oriana Bercea
09:30	*How to design algorithms for GPUs* Nodari Sitchinava
10:00	*Sparse Regular Expressions* Inge Li Gørtz
10:30	*Coffee break*
11:00	*Sorting pattern-avoiding permutations and forbidden 0-1 matrices* Seth Pettie
11:30	*Maplets and their Application* Rob Johnson
12:15	*Lunch*
15:30	*Coffee & Cake*
16:00	*Morphing Graph Drawings* Elena Arseneva
16:30	*Ordered Majority* Ian Munro
17:00	*Parameterized algorithm for the planar disjoint paths problem* Eunjin Oh
17:30	*Algorithms for Processing in Memory* Francesco Silvestri
18:00	*Dinner*

Friday	**May 26, 2023**
07:30	*Breakfast*
09:00	*Sorting with Priced Comparisons: The General, the Bichromatic, and the Universal* Riko Jacob
09:30	*Online List Labeling: Breaking the* $\log^2 n$ *Barrier* Nicole Wein
10:00	*Dynamic Filters and Retrieval with one Memory Access* Guy Even
10:30	*Conditional Lower Bounds for Dynamic Geometric Problems* Justin Dallant
11:00	*Coffee break*
12:15	*Lunch*

4 Overview of Talks

4.1 Should We Teach Parallelism throughout Undergraduate Algorithm Courses?

Guy E. Blelloch (Carnegie Mellon University – Pittsburgh, US)

4.2 Iceberg Hashing and Mosaic Pages

Alexander Conway (VMware Research – Palo Alto, US)

Joint work of Alexander Conway, Krishnan Gosakan, Jaehyun Han, William Kuszmaul, Ibrahim N. Mubarek, Nirjhar Mukherjee, Karthik Sriram, Guido Tagliavini, Evan West, Michael A. Bender, Abhishek Bhattacharjee, Martin Farach-Colton, Jayneel Gandhi, Rob Johnson, Sudarsun Kannan, Donald E. Porter

Main reference Krishnan Gosakan, Jaehyun Han, William Kuszmaul, Ibrahim N. Mubarek, Nirjhar Mukherjee, Karthik Sriram, Guido Tagliavini, Evan West, Michael A. Bender, Abhishek Bhattacharjee, Alex Conway, Martin Farach-Colton, Jayneel Gandhi, Rob Johnson, Sudarsun Kannan, Donald E. Porter: "Mosaic Pages: Big TLB Reach with Small Pages", in Proc. of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3, ASPLOS 2023, Vancouver, BC, Canada, March 25-29, 2023, pp. 433–448, ACM, 2023.

URL https://doi.org//10.1145/3582016.3582021

In this talk, I present an algorithmic approach to co-designing TLB hardware and the paging mechanism to increase TLB reach without the fragmentation issues incurred by huge pages. Along the way, I'll introduce a new hash-table design that overcomes existing tradeoffs, and achieves better performance than state-of-the-art hash tables both in theory and in practice. Key to these results are "tiny pointers," an algorithmic technique for compressing pointers.

4.3 Quicksort, Timsort, Powersort – Algorithmic ideas, engineering tricks, and trivia behind CPython's new sorting algorithm

Sebastian Wild (University of Liverpool, GB)

Main reference J. Ian Munro, Sebastian Wild: "Nearly-Optimal Mergesorts: Fast, Practical Sorting Methods That Optimally Adapt to Existing Runs", in Proc. of the 26th Annual European Symposium on Algorithms, ESA 2018, August 20-22, 2018, Helsinki, Finland, LIPIcs, Vol. 112, pp. 63:1–63:16, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018.
URL https://doi.org//10.4230/LIPIcs.ESA.2018.63

Writing a sorting function is easy – coding a fast and reliable reference implementation less so. In this talk, I tell the story behind CPython's latest updates of the list sort function.

After using Quicksort for a long while, Tim Peters invented Timsort, a clever Mergesort variant, for the CPython reference implementation of Python. Timsort is both effective in Python and a popular export product: it is used in many languages and frameworks, notably OpenJDK, the Android runtime, and the V8 JavaScript engine.

Despite this success, algorithms researchers eventually pinpointed two flaws in Timsort's underlying algorithm: The first could lead to a stack overflow in CPython (and Java); although it has meanwhile been fixed, it is curious that 10 years of widespread use didn't bring it to surface. The second flaw is related to performance: the order in which detected sorted segments, the "runs" in the input, are merged, can be 50% more costly than necessary. Based on ideas from the little known puzzle of optimal alphabetic trees, the Powersort merge policy finds nearly optimal merging orders with negligible overhead, and is now (Python 3.11.0) part of the CPython implementation.

References

1 J. Ian Munro and Sebastian Wild. *Nearly-Optimal Mergesorts: Fast, Practical Sorting Methods That Optimally Adapt to Existing Runs.* ESA 2018
2 William Cawley Gelling, Markus E. Nebel, Benjamin Smith, and Sebastian Wild. *Multiway Powersort.* ALENEX 2023

4.4 Scalable Data Structures for Privacy on Graphs

Quanquan C. Liu (Northwestern University – Evanston, US)

Joint work of Laxman Dhulipala, Quanquan C. Liu, Sofya Raskhodnikova, Jessica Shi, Julian Shun, Shangdi Yu
Main reference Laxman Dhulipala, Quanquan C. Liu, Sofya Raskhodnikova, Jessica Shi, Julian Shun, Shangdi Yu: "Differential Privacy from Locally Adjustable Graph Algorithms: k-Core Decomposition, Low Out-Degree Ordering, and Densest Subgraphs", in Proc. of the 63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 – November 3, 2022, pp. 754–765, IEEE, 2022.
URL https://doi.org//10.1109/FOCS54457.2022.00077
Main reference Talya Eden, Quanquan C. Liu, Sofya Raskhodnikova, Adam D. Smith: "Triangle Counting with Local Edge Differential Privacy", in Proc. of the 50th International Colloquium on Automata, Languages, and Programming, ICALP 2023, July 10-14, 2023, Paderborn, Germany, LIPIcs, Vol. 261, pp. 52:1–52:21, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023.
URL https://doi.org//10.4230/LIPIcs.ICALP.2023.52

Differential privacy is the gold standard for rigorous data privacy guarantees where the traditional central model assumes a trusted curator who takes private input and outputs privatized answers to user queries. However, one major assumption in this model is that the trusted curator always keeps the input private. Unfortunately, such a strong notion of trust is too ideal in today's world where massive data leaks occur. Thus, in this talk, I'll discuss private graph algorithms in the local model, where nodes trust no one with their private information. One major issue in the local privacy model for graphs is scalability. Scalability is often an issue for graph algorithms in the local privacy model because many techniques for making graph algorithms private change the density of the input graph (i.e. making the graph much denser than it was previously). I'll present the first local edge differentially private (LEDP) algorithms for k-core decomposition, low out-degree ordering, and densest subgraph. Furthermore, our algorithms are scalable and can be implemented in distributed and parallel models without the issue of changing graph density. Our algorithm's approximation factor matches that of the currently best non-private distributed algorithm for k-core decomposition with only an poly$(\log n)/\epsilon$ additive error. I'll conclude with a discussion of some open questions and potential future work.

4.5 Parallel Longest Increasing Subsequence and Van Emde Boas Trees

Yihan Sun (University of California – Riverside, US)

Joint work of Yan Gu, Ziyang Men, Zheqi Shen, Yihan Sun, Zijin Wan
Main reference Yan Gu, Ziyang Men, Zheqi Shen, Yihan Sun, Zijin Wan: "Parallel Longest Increasing Subsequence and van Emde Boas Trees", in Proc. of the 35th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA 2023, Orlando, FL, USA, June 17-19, 2023, pp. 327–340, ACM, 2023.
URL https://doi.org//10.1145/3558481.3591069

This paper studies parallel algorithms for the longest increasing subsequence (LIS) problem. Let n be the input size and k be the LIS length of the input. Sequentially, LIS is a simple problem that can be solved using dynamic programming (DP) in $O(n \log n)$ work. However, parallelizing LIS is a long-standing challenge. We are unaware of any parallel LIS algorithm that has optimal $O(n \log n)$ work and non-trivial parallelism (i.e., $\tilde{O}(k)$ or $o(n)$ span). This paper proposes a parallel LIS algorithm that costs $O(n \log k)$ work, $\tilde{O}(k)$ span, and $O(n)$ space, and is much simpler than the previous parallel LIS algorithms. We also generalize the algorithm to a weighted version of LIS, which maximizes the weighted sum for all objects in an increasing subsequence. To achieve a better work bound for the weighted LIS algorithm, we designed parallel algorithms for the van Emde Boas (vEB) tree, which has the same structure as the sequential vEB tree, and supports work-efficient parallel batch insertion, deletion, and range queries.

We also implemented our parallel LIS algorithms. Our implementation is light-weighted, efficient, and scalable.

4.6 Lomuto's comeback or the unpredictability of program efficiency

Markus E. Nebel (Universität Bielefeld, DE)

Joint work of Markus E. Nebel, David Komnick
Main reference https://dlang.org/blog/2020/05/14/lomutos-comeback/

We report experimental results for a Quicksort variant based on Lomuto's partitioning suggested by Andrei Alexandrescu. This variant eliminates branches inside the main loop of the partitioning process for the price of an increased number of overall executed instructions.

However, the resulting reduction of mispredicted branches gives rise to a heavily used pipeline. As a consequence, the resulting Quicksort implementation runs faster than one using Hoare/Sedgewick partitioning. Our adaptations of the latter to different branch free versions imply a similar overhead of instructions and a comparable reduction of branch misses. However, the speedup observed due to improved pipelining is way smaller than for the Lomuto variant and results in an overall worse runtime. So far we have no explanation for this.

4.7 Sliding Block Hashing (Slick)

Peter Sanders (KIT – Karlsruher Institut für Technologie, DE)

License Creative Commons BY 4.0 International license
© Peter Sanders
Joint work of Hans-Peter Lehmann, Peter Sanders, Stefan Walzer
Main reference Hans-Peter Lehmann, Peter Sanders, Stefan Walzer: "Sliding Block Hashing (Slick) – Basic Algorithmic Ideas", CoRR, Vol. abs/2304.09283, 2023.
URL https://doi.org//10.48550/arXiv.2304.09283

We present **Sli**ding Blo**ck** Hashing (Slick) a simple hash table data structure that combines high performance with very good space efficiency.

4.8 Online List Labeling with Predictions

Shikha Singh (Williams College – Williamstown, US)

License Creative Commons BY 4.0 International license
© Shikha Singh
Joint work of Samuel McCauley, Benjamin Moseley, Aidin Niaparast, Shikha Singh
Main reference Samuel McCauley, Benjamin Moseley, Aidin Niaparast, Shikha Singh: "Online List Labeling with Predictions", CoRR, Vol. abs/2305.10536, 2023.
URL https://doi.org//10.48550/arXiv.2305.10536

A growing line of work shows how learned predictions can be used to break through worst-cast barriers to improve the running time of an algorithm. However, incorporating predictions into data structures with strong theoretical guarantees remains underdeveloped. This work takes a step in this direction by showing that predictions can be leveraged in the fundamental online list labeling problem. In the problem, n items arrive over time and must be stored in sorted order in an array of size $\Theta(n)$. The array slot of an element is its label and the goal is to maintain sorted order while minimizing the total number of elements moved (i.e., relabeled). We present a new list labeling data structure and bound its performance in two models. In the worst-case learning-augmented model, we give guarantees in terms of the error in the predictions. Our data structure provides strong theoretical guarantees— it is optimal for any prediction error and guarantees the best-known worst-case bound even when the predictions are entirely erroneous. We also consider a stochastic error model and bound the performance in terms of the expectation and variance of the error. Finally, the theoretical results are demonstrated empirically. In particular, we show that our data structure performs well on numerous real datasets, including temporal data sets where predictions are constructed from elements that arrived in the past (as is typically done in a practical use case).

4.9 Simple Hash Tables

Martin Farach-Colton (Rutgers University – Piscataway, US)

We present a hash table that achieves nearly the state of the art but requires only basic analytical techniques. The aim is to present an algorithm that can be taught to graduate students (or perhaps advanced undergrads).

4.10 Write-Optimized Algorithms for Stream Monitoring

Cynthia A. Phillips (Sandia National Labs – Albuquerque, US)

Joint work of Shikha Singh, Prashant Pandey, Michael Bender, Jonathan Berry, Daniel Delayo, Martin Farach-Colton, Rob Johnson, Thomas Kroeger, Cynthia Phillips, David Tench, Eric Thomas
Main reference Shikha Singh, Prashant Pandey, Michael A. Bender, Jonathan W. Berry, Martin Farach-Colton, Rob Johnson, Thomas M. Kroeger, Cynthia A. Phillips: "Timely Reporting of Heavy Hitters Using External Memory", ACM Trans. Database Syst., Vol. 46(4), pp. 14:1–14:35, 2021.
URL https://doi.org//10.1145/3472392

We describe data structures and data-management algorithms for monitoring cyber streams. We wish to identify specific patterns that arrive slowly over time, hidden among high-speed streams of normal traffic. The key piece of the Firehose benchmark that models this application is a variant of the heavy-hitters problem: report a key after it has been seen a specific constant number of times. To solve this without false negatives requires $\Omega(N)$ space for partial-pattern storage for a stream of size N. We give write-optimized external-memory algorithms to accurately monitor high-speed streams with provable tunable trade-off between reporting delay and I/O overhead. Our experimental results show that a multithreaded version of our algorithm has throughput comparable to an engineered in-RAM reference implementation, but our method reports all reportable keys, while the in-RAM method can miss almost all reports for sufficiently large key space. We describe extensions to unending streams.

4.11 Optimizing Search Layouts in Packed Memory Arrays

Helen Xu (Lawrence Berkeley National Laboratory, US)

Joint work of Brian Wheatman, Randal C. Burns, Aydin Buluç, Helen Xu
Main reference Brian Wheatman, Randal C. Burns, Aydin Buluç, Helen Xu: "Optimizing Search Layouts in Packed Memory Arrays", in Proc. of the Symposium on Algorithm Engineering and Experiments, ALENEX 2023, Florence, Italy, January 22-23, 2023, pp. 148–161, SIAM, 2023.
URL https://doi.org//10.1137/1.9781611977561.ch13

This talk covers Search-optimized Packed Memory Arrays (SPMAs), a collection of data structures based on Packed Memory Arrays (PMAs) that address suboptimal search via cache-optimized search layouts. Traditionally, PMAs and B-trees have tradeoffs between searches/inserts and scans: B-trees were faster for searches and inserts, while PMAs were faster for scans. Our empirical evaluation shows that SPMAs overcome this tradeoff for unsorted input distributions: on average, SPMAs are faster than B+-trees (a variant of B-trees optimized for scans) on all major operations.

4.12 Using Data Structures within Parallel Algorithms

Kunal Agrawal (Washington University – St. Louis, US)

4.13 Dynamic Spanners

Rasmus Kyng (ETH Zürich, CH)

Joint work of Li Chen, Yang Liu, Richard Peng, Maximilian Probst Gutenberg, Sushant Sachdeva
Main reference Li Chen, Rasmus Kyng, Yang P. Liu, Richard Peng, Maximilian Probst Gutenberg, Sushant Sachdeva: "Maximum Flow and Minimum-Cost Flow in Almost-Linear Time", CoRR, Vol. abs/2203.00671, 2022.
URL https://doi.org//10.48550/arXiv.2203.00671

I gave a presentation on two topics:
1. How to use an L1 IPM to turn the task of solving an LP into a sequence data structure queries, and how this can be applied to solve minimum-cost flow problems via min-ratio cycle updates.
2. How to design dynamic spanners that allow for edge insertions and deletions and vertex splits.

4.14 Batch Parallel fast Worst Case Dynamic Connectivity

Valerie King (University of Victoria, CA)

The dynamic connectivity problem is to process an online sequence of edge insertions and deletions in a graph while answering connectivity queries. Here we simplify and parallelize the sequential Monte Carlo algorithms of King et al. as improved by Wang for dynamic connectivity, which requires polylogarithmic time per update and per query in the worst-case. Our simplification adds no cost to the asymptotic sequential running time. It enables us to rely strictly on ET-trees, rather than more complicated path compression data structures, making it simpler to perform batch-parallel updates.

4.15 Sparsity-adaptive dynamic graph algorithms

Eva Rotenberg (Technical University of Denmark – Lyngby, DK)

Joint work of Aleksander B. G. Christiansen, Jacob Holm, Ivor van der Hoog, Krzysztof Nowicki, Eva Rotenberg, Chris Schwiegelshohn, Carsten Thomassen

The *arboricity* α of a graph is the number of forests it takes to cover all its edges. Being asymptotically related to the graph's degeneracy and maximal subgraph density, arboricity is considered a good measure of the sparsity of a graph. Natural computational questions about arboricity include: computing the arboricity, obtaining a decomposition of the edges into few forests, and orienting the edges so that each vertex has only close to α out-edges.

In this talk, we will address these questions in the *dynamic* setting, in which the graph is subject to arbitrary, adversarial insertions and deletions of edges. We will see how maintaining an orientation with few out-edges from each vertex leads to efficient dynamic algorithms for *matching*, *colouring*, and decomposing into $O(\alpha)$ forests; And we will see how to efficiently balance the number of out-edges in an orientation of the dynamic graph, via an almost local reconciliation between neighbouring vertices.

References

1 Aleksander B. G. Christiansen, Jacob Holm, Eva Rotenberg, and Carsten Thomassen. On Dynamic $\alpha + 1$ Arboricity Decomposition and Out-Orientation. In Stefan Szeider, Robert Ganian, and Alexandra Silva, editors, *47th MFCS*, volume 241 of *LIPIcs*, pages 34:1–34:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.

2 Aleksander B. G. Christiansen, Jacob Holm, Ivor van der Hoog, Eva Rotenberg, and Chris Schwiegelshohn. Adaptive Out-Orientations with Applications. *CoRR*, abs/2209.14087, 2022.

3 Aleksander Bjørn Grodt Christiansen, Krzysztof Nowicki, and Eva Rotenberg. Improved Dynamic Colouring of Sparse Graphs. In Barna Saha and Rocco A. Servedio, editors, *Proceedings of the 55th Annual STOC*, pages 1201–1214. ACM, 2023.

4 Aleksander B. G. Christiansen and Eva Rotenberg. Fully-Dynamic $\alpha + 2$ Arboricity Decompositions and Implicit Colouring. In Mikolaj Bojanczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th ICALP 2022*, volume 229 of *LIPIcs*, pages 42:1–42:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.

4.16 The k-Dyck Edit Distance Problem

Tsvi Kopelowitz (Bar-Ilan University – Ramat Gan, IL)

Joint work of Dvir Fried, Shay Golan, Tomasz Kociumaka, Tsvi Kopelowitz, Ely Porat, Tatiana Starikovskaya
Main reference Dvir Fried, Shay Golan, Tomasz Kociumaka, Tsvi Kopelowitz, Ely Porat, Tatiana Starikovskaya: "An Improved Algorithm for The k-Dyck Edit Distance Problem", in Proc. of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 – 12, 2022, pp. 3650–3669, SIAM, 2022.
URL https://doi.org//10.1137/1.9781611977073.144

A Dyck sequence is a sequence of opening and closing parentheses (of various types) that is balanced. The Dyck edit distance of a given sequence of parentheses S is the smallest number of edit operations (insertions, deletions, and substitutions) needed to transform S into a Dyck sequence. We consider the threshold Dyck edit distance problem, where the input is a sequence of parentheses S and a positive integer k, and the goal is to compute the Dyck edit distance of S only if the distance is at most k, and otherwise report that the distance is larger than k. Backurs and Onak [PODS'16] showed that the threshold Dyck edit distance problem can be solved in $O(n + k^{16})$ time. In this work, we design new algorithms for the threshold Dyck edit distance problem which costs $O(n + k^{4.782036})$ time with high probability or $O(n + k^{4.853059})$ deterministically. Our algorithms combine several new structural properties of the Dyck edit distance problem, a refined algorithm for fast (min, +) matrix product, and a careful modification of ideas used in Valiant's parsing algorithm.

4.17 Heaps

Robert E. Tarjan (Princeton University, US)

The heap, or priority queue data structure supports fast access to the smallest of a set of items subject to insertions, deletions, and decreases in value. We describe recent results giving tight and almost-tight amortized efficiency bounds for three self-adjusting heap implementations, the slim heap, the smooth heap, and the multipass pairing heap.

4.18 An Extendable Data Structure for Incremental Stable Perfect Hashing

Ioana Oriana Bercea (IT University of Copenhagen, DK)

Joint work of Ioana Oriana Bercea, Guy Even

Main reference Ioana Oriana Bercea, Guy Even: "An extendable data structure for incremental stable perfect hashing", in Proc. of the STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 – 24, 2022, pp. 1298–1310, ACM, 2022.
URL https://doi.org//10.1145/3519935.3520070

The talk is about recent advancements in the design of dictionaries and other related data structures. A dynamic dictionary is a data structure that maintains sets under insertions and deletions and supports membership queries of the form "is an element x in the set or not?". A related problem is that of maintaining a perfect hash function over the set, in which the data structure assigns a unique hashcode to each element in the set (but does not need to support membership queries). We specifically focus on showing a perfect hashing data structure whose space is proportional to the cardinality of the set at all points in time and whose hashcodes are stable in the incremental setting (i.e., when only insertions are allowed, the hashcode of an element does not change while the element is in the set). This is joint work with Guy Even, based on our STOC 2022 paper.

4.19 How to design algorithms for GPUs

Nodari Sitchinava (University of Hawaii at Manoa – Honolulu, US)

In this talk we will briefly review the models of computation used for designing algorithms on GPUs. We will also discuss recent algorithmic results on how to avoid bank conflicts when designing algorithms for GPUs.

4.20 Sparse Regular Expression Matching

Inge Li Gørtz (Technical University of Denmark – Lyngby, DK)

Joint work of Philip Bille, Inge Li Gørtz
Main reference Philip Bille, Inge Li Gørtz: "Sparse Regular Expression Matching", CoRR, Vol. abs/1907.04752, 2019.
URL http://arxiv.org/abs/1907.04752

A regular expression specifies a set of strings formed by single characters combined with concatenation, union, and Kleene star operators. Given a regular expression R and a string Q, the regular expression matching problem is to decide if Q matches any of the strings specified by R. Regular expressions are a fundamental concept in formal languages and regular expression matching is a basic primitive for searching and processing data. A standard textbook solution [Thompson, CACM 1968] constructs and simulates a nondeterministic finite automaton, leading to an $O(nm)$ time algorithm, where n is the length of Q and m is the length of R. Despite considerable research efforts only polylogarithmic improvements of this bound are known. Recently, conditional lower bounds provided evidence for this lack of progress when Backurs and Indyk [FOCS 2016] proved that, assuming the strong exponential time hypothesis (SETH), regular expression matching cannot be solved in $O((nm)^{1-\epsilon})$, for any constant $\epsilon > 0$. Hence, the complexity of regular expression matching is essentially settled in terms of n and m.

In this paper, we take a new approach and go beyond worst-case analysis in n and m. We introduce a *density* parameter, Δ, that captures the amount of nondeterminism in the NFA simulation on Q. The density is at most $nm + 1$ but can be significantly smaller. Our main result is a new algorithm that solves regular expression matching in

$$O\left(\Delta \log\log \frac{nm}{\Delta} + n + m\right)$$

time.

This essentially replaces nm with Δ in the complexity of regular expression matching. We complement our upper bound by a matching conditional lower bound that proves that we cannot solve regular expression matching in time $O(\Delta^{1-\epsilon})$ for any constant $\epsilon > 0$ assuming SETH.

4.21 Sorting Pattern-avoiding Permutations and Forbidden 0-1 Matrices

Seth Pettie (University of Michigan – Ann Arbor, US)

Joint work of Parinya Chalermsook, Seth Pettie, Sorrachai Yingchareonthawornchai
Main reference Parinya Chalermsook, Seth Pettie, Sorrachai Yingchareonthawornchai: "Forbidden 0-1 Matrices and the Complexity of Sorting Pattern-Avoiding Permutations". Manuscript (2023).

An n-permutation S "avoids" a k-permutation π if there are no k indices $i_1 < \cdots < i_k$ such that $S(i_j) < S(i_{j'})$ iff $\pi(j) < \pi(j')$. Chalermsook et al. (2015) and Kozma and Saranurak (2019) presented two algorithms for sorting such permutations in $O(n \cdot 2^{(\alpha(n))^{3k/2}})$ time. Their upper bound was derived by transcribing the behavior of the algorithm as an $n \times n$ 0-1 matrix and proving that this matrix avoids a certain pattern, which is the Kronecker product of a permutation (encoding π) and a "hat" pattern. In this talk I prove that both of these algorithms actually run in $O(n \cdot 2^{(1+o(1))\alpha(n)})$ time, by bounding the extremal function of all such patterns. I also show that this bound is essentially tight, and that most such patterns have an extremal function that is $\Omega(n \cdot 2^{\alpha(n)})$.

4.22 Maplets and their Application

Rob Johnson (VMware – Palo Alto, US)

Filters, such as Bloom, quotient, cuckoo, xor, and ribbon filters, are space-efficient, lossy representations of sets. They have become widely used in systems and extensively researched by data structures designers. In this talk, we argue that most applications would be better served by maplets, i.e. space efficient, lossy maps, rather than filters. We explain how to generalize the filter notion of lossiness to maps, show how several classic filter applications can be dramatically improved by using maplets, and show how to construct maplets straightforwardly from many of today's filters.

4.23 Morphing planar drawings of graphs: main results, morphing queries and a progress report

Elena Arseneva (University of Lugano, CH)

Joint work of Elena Arseneva, Eunjin Oh

Given two planar (straight-line) drawings of a planar graph, can one drawing be transformed to the other in a little number of steps, where during each step every vertex moves along a straight line segment with a uniform speed? It is crucially required that there is no crossing between any elements of a drawing at any moment. Such transformation is called a morph. If the drawings are topologically equivalent, then a 2D morph is always possible in $O(n)$ steps, and sometimes a linear number of steps is necessary [1]. Allowing intermediate drawings to lie in 3D reduces the upper bound on the number of steps to $O(\log n)$ for trees [2], and lifts the topologically equivalent requirement for arbitrary graphs, however at a cost of quadratic number of steps [3]. No non-trivial lower bound for this setting is known. During the seminar I proposed a query variant of the graph morphing problem, and jointly with Eunjin Oh we obtained the first result in this direction:

A planar drawing Gamma of an n-vertex graph G can be preprocessed in $O(n^{5/3} \log^2 n)$ expected time and $O(n \log n)$ space for the following queries: given a vertex v of G, and a point t in R^2, can v be moved from its position in Gamma to the point t, such that the morph is non-crossing?

After the above preprocessing, this query can be answered in $O(\log m \log^2 n)$ time, when m is the degree of vertex v. We hope to further improve this runtime and generalise our data structure to availability region queries, updates and 3D morphs.

This result is a joint work during the seminar with its participant Eunjin Oh.

References

1 S. Alamdari, P. Angelini, F. Barrera-Cruz, T. Chan, G. Da Lozzo, G. Di Battista, F. Frati, P. Haxell, A. Lubiw, M. Patrignani, V. Roselli How to morph planar graph drawings. *SIAM Journal on Computing*, 46(2):824-52, 2017.

2 E. Arseneva, P. Bose, P. Cano, A. D'Angelo, V. Dujmovic, F. Frati, S. Langerman, and A. Tappini. Pole Dancing: 3D Morphs for Tree Drawings. *Journal of Graph Algorithms and Applications (Special issue of selected papers from GD'18)*, 23(3), pages 579–602 DOI: 10.7155/jgaa.00503, 2019.

3 K. Buchin, W. Evans, F. Frati, I. Kostitsyna, M. Löffler, T. Ophelders, A. Wolff. Morphing planar graph drawings through 3d. In *Proc. International Conference on Current Trends in Theory and Practice of Computer Science 2023 Jan 1 (pp. 80-95).*

4.24 The Ordered Majority Problem

Ian Munro (University of Waterloo, CA)

Joint work of Louisa Seelbach Benkner, Ben Baral, Siwei Yang, Ian Munro

We examine the well-known problem of determining whether some value occurs in the majority of positions in an array. The twist is that in the past comparisons have been $(=, \neq)$, while here we focus on a three way outcome $(<, =, >)$. We show, that like the $(=, \neq)$ version, $3n/2 - o(1)$ comparisons are necessary (and sufficient) in the worst case, but give a Las Vegas algorithm requiring an expected $n + o(1)$ comparisons and $n - o(n)$ lower bound, in contrast with a $1.059...n$ lower bound for the $(=, \neq)$ version.

4.25 Parameterized algorithm for the planar disjoint paths problem

Eunjin Oh (POSTECH – Pohang, KR)

Joint work of Kyungjin Cho, Eunjin Oh, Seunghyeok Oh
Main reference Kyungjin Cho, Eunjin Oh, Seunghyeok Oh: "Parameterized Algorithm for the Disjoint Path Problem on Planar Graphs: Exponential in k^2 and Linear in n", in Proc. of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023, pp. 3734–3758, SIAM, 2023.
URL https://doi.org//10.1137/1.9781611977554.ch144

Given a planar graph G with n vertices and a set $T = \{(s_1, t_1), \dots, (s_k, t_k)\}$ of k terminal pairs, the disjoint paths problem asks for computing a set of vertex disjoint paths $P_1, \dots, P_k$ such that P_i connects s_i and t_i. In this talk, I will introduce a $2^{O(k^2)}n$-time algorithm for the planar disjoint paths problem. This improves the previously best-known algorithm with running times of $2^{O(k^2)}n^6$ and $2^{2^{O(k)}}n$.

4.26 Algorithms for Processing-In-Memory

Francesco Silvestri (University of Padova, IT)

Joint work of Lorenzo Asquini, Juan Gomez-Luna, Francesco Silvestri

Processing-In-Memory (PIM) is a hardware architecture that allows reducing the memory bottleneck: while data are sent from the memory to the CPU in the traditional memory hierarchy, in a PIM architecture the computation is sent to the memory. The main idea is to add a small computing unit within each memory module: this is an almost 40-year-old theoretical idea; however, only recently PIM architectures have been successfully implemented and commercialized (e.g., by UPMEM). In this talk, we will see a computational model for designing efficient algorithms that fully exploit PIMs, and introduce PIM algorithms for binary search and triangle counting.

4.27 Sorting with Priced Comparisons: The General, the Bichromatic, and the Universal

Riko Jacob (IT University of Copenhagen, DK)

Joint work of Mayank Goswami, Riko Jacob
Main reference Mayank Goswami, Riko Jacob: "Universal Sorting: Finding a DAG using Priced Comparisons", CoRR, Vol. abs/2211.04601, 2022.
URL https://doi.org//10.48550/arXiv.2211.04601

We address two open problems in sorting with priced information, introduced by [Charikar, Fagin, Guruswami, Kleinberg, Raghavan, Sahai (CFGKRS), STOC 2000]. In this setting, different comparisons have different (potentially infinite) costs. The goal is to find a sorting algorithm with small competitive ratio, defined as the (worst-case) ratio of the algorithm's cost to the cost of the cheapest proof of the sorted order.

1) When all costs are in $\{0, 1, n, \infty\}$, we give an algorithm that has $\widetilde{O}(n^{3/4})$ competitive ratio. Our result refutes the hypothesis that a widely cited $\Omega(n)$ lower bound on the competitive ratio for finding the maximum extends to sorting. This lower bound by [Gupta, Kumar, FOCS 2000] uses costs in $\{0, 1, n, \infty\}$ and was claimed as the reason why sorting with arbitrary costs seemed bleak and hopeless. Our algorithm also generalizes the algorithms for generalized sorting (all costs in $\{1, \infty\}$), a version initiated by [Huang, Kannan, Khanna, FOCS 2011] and addressed recently by [Kuszmaul, Narayanan, FOCS 2021].

2) We answer the problem of bichromatic sorting posed by [CFGKRS]: We are given two sets A and B of total size n, and the cost of an $A - A$ comparison or a $B - B$ comparison is higher than an $A - B$ comparison. The goal is to sort $A \cup B$. An $\Omega(\log n)$ lower bound on competitive ratio follows from unit-cost sorting. We give a randomized algorithm with an almost-optimal w.h.p. competitive ratio of $O(\log^3 n)$.

We also study generalizations of the problem *universal sorting* and *bipartite sorting* (a generalization of nuts-and-bolts). Here, we define a notion of *instance optimality*, and develop an algorithm for bipartite sorting which is $O(\log^3 n)$ instance-optimal. Our framework of instance optimality applies to other static problems and may be of independent interest.

4.28 Online List Labeling: Breaking the $\log^2 n$ Barrier

Nicole Wein (Rutgers University – Piscataway, US)

Joint work of Michael Bender, Alexander Conway, Martin Farach-Colton, Hanna Komlos, William Kuszmaul, Nicole Wein
Main reference Michael A. Bender, Alexander Conway, Martin Farach-Colton, Hanna Komlós, William Kuszmaul, Nicole Wein: "Online List Labeling: Breaking the $\log^2$n Barrier", CoRR, Vol. abs/2203.02763, 2022.
URL https://doi.org//10.48550/arXiv.2203.02763

The online list labeling problem is a basic primitive in data structures. The goal is to store a dynamically-changing set of n items in an array of m slots, while keeping the elements in sorted order. To do so, some items may need to be moved over time, and the goal is to minimize the number of items moved per insertion/deletion. When $m = Cn$ for some constant $C > 1$, an upper bound of $O(\log^2 n)$ items moved per insertion/deletion has been known since 1981. There is a matching lower bound for deterministic algorithms, but for randomized algorithms, the best known lower bound is $\Omega(\log n)$, leaving a gap between upper and lower bounds. We improve the upper bound, providing a randomized data structure with expected $O(\log^{3/2} n)$ items moved per insertion/deletion.

4.29 Dynamic Filter and Retrieval with One Memory Access

Guy Even (Tel Aviv University, IL)

Joint work of Ioana-Oriana Bercea, Guy Even, Tomer Even

We present two dynamic data-structures in the word RAM model. The first data structure is a filter that supports approximate membership queries that is characterized by the following properties:

1. Dynamic: supports insert, delete, and membership-query operations.
2. Can store a dataset of cardinality at most n (insertions may fail with probability $o(1/\text{poly}(n))$).
3. Adjustable false-positive probability ε, provided that $\varepsilon = \Omega(1/(\text{polylog } n))$.
4. Space-efficient: $(1 + o(1)) \cdot n \log_2(1/\epsilon) + O(n)$ bits.
5. Worst case constant time for every operation.
6. Single memory access per operation (not including hash function evaluation).

The second data structure is a retrieval data structure that supports value queries and is characterized by the following properties:

1. Supports the following operations: insert key-value pairs, delete a key, and query the value of a key.
2. Can store a set of at most n key-value pairs (insertions may fail with probability $1/\text{poly}(n)$).
3. Values are binary strings of length $O(\log \log n)$ bits.
4. Space-compact: $O(n \log \log n)$ bits.
5. Worst case constant time for every operation.
6. The expected number of memory accesses per operation is $1 + 1/\text{polylog}(n)$ (not including hash function evaluation).

The retrieval data structure has an additional "false-positive" feature: the response to a query for a key not in the dataset is NULL with probability at least $1 - 1/\text{polylog}(n)$.

4.30 Conditional Lower Bounds for Dynamic Geometric Measure Problems

Justin Dallant (UL – Brussels, BE)

Joint work of Justin Dallant, John Iacono
Main reference Justin Dallant, John Iacono: "Conditional Lower Bounds for Dynamic Geometric Measure Problems", in Proc. of the 30th Annual European Symposium on Algorithms, ESA 2022, September 5-9, 2022, Berlin/Potsdam, Germany, LIPIcs, Vol. 244, pp. 39:1–39:17, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.
URL https://doi.org//10.4230/LIPIcs.ESA.2022.39

We give new polynomial lower bounds for a number of dynamic measure problems in computational geometry. These lower bounds hold in the Word-RAM model, conditioned on the hardness of either 3SUM, APSP, or the Online Matrix-Vector Multiplication problem [Henzinger et al., STOC 2015]. In particular we get lower bounds in the incremental and fully-dynamic settings for counting maximal or extremal points in $\mathbb{R}^3$, different variants of Klee's Measure Problem, problems related to finding the largest empty disk in a set of points, and querying the size of the i'th convex layer in a planar set of points. We also answer a question of Chan et al. [SODA 2022] by giving a conditional lower bound for dynamic approximate square set cover. While many conditional lower bounds for dynamic data structures have been proven since the seminal work of Patrascu [STOC 2010], few of them relate to computational geometry problems. This is the first paper focusing on this topic. Most problems we consider can be solved in $O(n \log n)$ time in the static case and their dynamic versions have only been approached from the perspective of improving known upper bounds. One exception to this is Klee's measure problem in $\mathbb{R}^2$, for which Chan [CGTA 2010] gave an unconditional $\Omega(\sqrt{n})$ lower bound on the worst-case update time. By a similar approach, we show that such a lower bound also holds for an important special case of Klee's measure problem in $\mathbb{R}^3$ known as the Hypervolume Indicator problem, even for amortized runtime in the incremental setting.

Participants

- Kunal Agrawal
Washington University – St. Louis, US
- Elena Arseneva
University of Lugano, CH
- Michael A. Bender
Stony Brook University, US
- Ioana Oriana Bercea
IT University of Copenhagen, DK
- Guy E. Blelloch
Carnegie Mellon University – Pittsburgh, US
- Gerth Stølting Brodal
Aarhus University, DK
- Alexander Conway
VMware Research – Palo Alto, US
- Justin Dallant
UL – Brussels, BE
- Guy Even
Tel Aviv University, IL
- Rolf Fagerberg
University of Southern Denmark – Odense, DK
- Martin Farach-Colton
Rutgers University – Piscataway, US
- Inge Li Gørtz
Technical University of Denmark – Lyngby, DK
- John Iacono
UL – Brussels, BE
- Riko Jacob
IT University of Copenhagen, DK
- Rob Johnson
VMware – Palo Alto, US
- Valerie King
University of Victoria, CA
- Tsvi Kopelowitz
Bar-Ilan University – Ramat Gan, IL
- László Kozma
FU Berlin, DE
- Rasmus Kyng
ETH Zürich, CH
- Moshe Lewenstein
Bar-Ilan University – Ramat Gan, IL
- Quanquan C. Liu
Northwestern University – Evanston, US
- Ulrich Carsten Meyer
Goethe-Universität – Frankfurt am Main, DE
- Ian Munro
University of Waterloo, CA
- Markus E. Nebel
Universität Bielefeld, DE
- Eunjin Oh
POSTECH – Pohang, KR
- Rotem Oshman
Tel Aviv University, IL
- Seth Pettie
University of Michigan – Ann Arbor, US
- Cynthia A. Phillips
Sandia National Labs – Albuquerque, US
- Vijaya Ramachandran
University of Texas – Austin, US
- Rajeev Raman
University of Leicester, GB
- Eva Rotenberg
Technical University of Denmark – Lyngby, DK
- Peter Sanders
KIT – Karlsruher Institut für Technologie, DE
- Francesco Silvestri
University of Padova, IT
- Shikha Singh
Williams College – Williamstown, US
- Nodari Sitchinava
University of Hawaii at Manoa – Honolulu, US
- Yihan Sun
University of California – Riverside, US
- Robert Endre Tarjan
Princeton University, US
- Nicole Wein
Rutgers University – Piscataway, US
- Sebastian Wild
University of Liverpool, GB
- Helen Xu
Lawrence Berkeley National Laboratory, US