Distant Representatives for Rectangles in the Plane

Therese Biedl
David R. Cheriton School of Computer Science, University of Waterloo, Canada

Anna Lubiw
David R. Cheriton School of Computer Science, University of Waterloo, Canada

Anurag Murty Naredla
David R. Cheriton School of Computer Science, University of Waterloo, Canada

Peter Dominik Ralbovsky
David R. Cheriton School of Computer Science, University of Waterloo, Canada

Graeme Stroud
David R. Cheriton School of Computer Science, University of Waterloo, Canada

Abstract

The input to the distant representatives problem is a set of n objects in the plane and the goal is to find a representative point from each object while maximizing the distance between the closest pair of points. When the objects are axis-aligned rectangles, we give polynomial time constant-factor approximation algorithms for the L_1, L_2, and L_∞ distance measures. We also prove lower bounds on the approximation factors that can be achieved in polynomial time (unless P = NP).

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Distant representatives, blocker shapes, matching, approximation algorithm, APX-hardness

Digital Object Identifier 10.4230/LIPIcs.ESA.2021.17

Supplementary Material Full version appeared at ArXiV.
Full Version: `https://arxiv.org/abs/2108.07751`

Funding *Therese Biedl*: Supported by NSERC.
Anna Lubiw: Supported by NSERC.

Acknowledgements We thank Jeffrey Shallit for help with continued fractions, and we thank anonymous reviewers for their helpful comments.

1 Introduction

The *distant representatives problem* was first introduced by Fiala et al. [17]. The name is a play-on-words on the term “distinct representatives” from Philip Hall’s classic work on bipartite matching [21]. The input is a set of geometric objects in a metric space. The goal is to choose one “representative” point in each object such that the points are distant from each other – more precisely, the objective is to maximize the distance between the closest pair of representative points. In the decision version of the problem, we are given a bound δ and the question is whether we can choose one representative point in each object such that the distance between any two points is at least δ.

The distant representatives problem has applications to map labelling and data visualization. To attach a label to each object, we can find representative points that are at least distance δ apart, and label each object with a ball of diameter δ (a square in L_∞) centred at its representative point.

The distant representatives problem is closely related to dispersion and packing problems. When all the objects are copies of a single object, the distant representatives problem becomes the dispersion problem: to choose k points in a region R to maximize the minimum distance between any two chosen points [3]. Equivalently, the problem is to pack k disjoint discs (in the chosen metric) of diameter δ into an expanded region and maximize δ. The distant representatives problem is also related to problems of "imprecise points" where standard computational geometry problems are solved when each input point is only known to lie within some small region [28].

There is a polynomial time algorithm for the distant representatives problem when the objects are segments on a line [32]. This result comes from the scheduling literature – each representative point is regarded as the centre-point of a unit length job. However, as shown by Fiala et al. [17], the decision version of distant representatives becomes NP-hard in 2D when the objects are unit discs for the L_2 norm or unit squares for the L_∞ norm.

Cabello [5] was the first to consider the optimization version of the distant representatives problem. He gave polynomial time approximation algorithms for the cases in 2D where the objects are squares under the L_∞ norm, or discs under the L_2 norm. The squares/discs may intersect and may have different sizes. His algorithms achieve an approximation factor of 2 in L_∞ and $\frac{8}{3}$ in L_2, with an improvement to 2.24 if the input discs are disjoint. A main idea in his solution is an "approximate-placement" algorithm that chooses representative points from a fine-enough grid using a matching algorithm; small squares/discs that do not contain grid points are handled separately. Cabello noted that the NP-hardness proof of Fiala et al. [17] can be modified to prove that there is no polynomial time approximation scheme (PTAS) for these problems unless P=NP. However, no one has given exact lower bounds on the approximation factors that can be achieved in polynomial time.

Our Results

We consider the distant representatives problem for axis-parallel rectangles in the plane. Rectangles are more versatile than squares or circles in many applications, e.g., for labelling rectangular Euler or Venn diagrams [29].

We give polynomial time approximation algorithms to find representative points for the rectangles such that the distance between any two representative points is at least $1/f$ times the optimum. The approximation factors f are given in Table 1 for the L_1, L_2, and L_∞ norms. Since rectangles are not fat objects [8], Cabello's approach of discretizing the problem by choosing representative points from a grid does not extend. Instead, we introduce a new technique of "imprecise discretization" and choose representative points from 1-dimensional shapes (e.g., +-shapes) arranged in a grid. After that, our plan is similar to Cabello's. First we solve an approximation version of the decision problem – to find representative points so long as the given distance δ is not too large compared to the optimum δ^*. Then we perform a search to find an approximation to δ^*. Unlike previous algorithms which use the real-RAM model, we use the word-RAM model, and thus must address bit complexity issues.

We accompany these positive results with lower bounds on the approximation factors that can be achieved in polynomial time (assuming P $\neq$ NP). The lower bounds are shown in Table 1. They apply even in the special case of horizontal and vertical line segments in the plane. The results are proved via gap-producing reductions from Monotone Rectilinear Planar 3-SAT [10]. These are the first explicit lower bounds on approximation factors for the distant representatives problem for any type of object.

Finally, we consider the even more special case of unit-length horizontal line segments, and the decision version of distant representatives. This is even closer to the tractable case of line segments on a line. However, Roeloffzen in his Master's thesis [30] proved NP-hardness for the L_2 norm. We give a more careful proof that takes care of bit complexity issues, and we show that the problem is NP-complete in the L_1 and L_∞ norms.

Table 1 Bounds on polynomial time approximation factors for the distant representatives problem for axis-aligned rectangles in the plane. A lower bound of x means that an approximation factor less than x implies P = NP. (For other L_p norms, there are some constant factors, but we have not optimized them.)

	L_1	L_2	L_∞
upper bound	5	$\sqrt{34} \approx 5.83$	6
lower bound	1.5	1.4425	1.5

For our algorithms and our hardness results, we must deal with bit complexity issues. For rectangles under the L_1 and L_∞ norms, we show that both the optimum value δ^* and the coordinates of an optimum solution have polynomially-bounded bit complexity. In particular, the decision problems lie in NP. The L_2 norm remains more of a mystery, and the decision problem can only be placed in $\exists\mathbb{R}$ (for an explanation of this class, see [6]).

Background

In one dimension, the decision version of the distant representatives problem for intervals on a line was solved by Barbara Simons [32], as a scheduling problem of placing disjoint unit jobs in given intervals. To transform the decision version of distant representatives to the scheduling problem, scale so $\delta = 1$, then expand each interval by $1/2$ on each side. The midpoints of the unit jobs provide the desired solution. Simons's decision algorithm was speeded up to $O(n \log n)$ by Garey et al. [20]. The optimum δ^* can be found using a binary search – in fact there is a discrete set of $O(n^3)$ possible δ^* values, which provides an $O(n^3 \log n)$ algorithm. (We see how to improve this to $O(n^2 \log n)$ but we are not aware of any published improvement.) There has been recent work on the online version of the problem [9]. The (offline) problem is easier when the intervals are disjoint. More generally, the problem is easier when the ordering of the representative points is specified, or is determined – for example if no interval is contained in another then there is an optimum solution where the ordering of the representative points is the same as the ordering of the interval's left endpoints. This "dispersion problem for ordered intervals" can be solved in linear time [26]. In a companion paper to this one, we improved this to a simpler algorithm using shortest paths in a polygon that solves the harder problem of finding the lexicographic maximum list of distances between successive pairs [4].

Cabello [5] gave polynomial time approximation algorithms for the distant representatives problem for balls in the plane, specifically for squares in L_∞ and for discs in L_2, with approximation factors of 2 and $\frac{8}{3} = 2.6\dot{6}$, respectively. For disjoint discs in L_2 he improved the approximation factor to 2.24. Dumitrescu and Jiang [14] further improved the approximation factor for disjoint discs to 1.414 (= 1/.707) by adding LP-based techniques to Cabello's approach. They also considered the case of unit discs, where they gave an algorithm with approximation factor 2.14 (= 1/.4674). For disjoint unit discs they gave a very simple algorithm with approximation factor 1.96 (= 1/.511). In a follow-up paper Dumitrescu and Jiang [15] gave bounds on the optimum δ^* for balls and cubes in L_2 depending on the minimum area of the union of subsets of k objects – these results have the flavour of Hall's classic condition for the existence of a set of distinct representatives.

The geometric dispersion problem (when all objects are copies of one object) was studied by Bauer and Fekete [3]. They considered the problem of placing k points in a rectilinear polygon with holes to maximize the min L_∞ distance between any two points or between a point and the boundary of the region. Equivalently, the problem is to pack k as-large-as-possible identical squares into the region. They gave a polynomial time 3/2-approximation algorithm, and proved that 14/13 is a lower bound on the approximation factor achievable in polynomial time. By contrast, if the goal is to pack as many squares of a given size into a region, the famous shifting-grid strategy of Hochbaum and Maas [23] provides a PTAS. Bauer and Fekete use this PTAS to design an approximate decision algorithm for their problem.

It is NP-hard to decide whether a square can be packed with given (different sized) squares [25] or discs [11]. For algorithmic approaches, see the survey [22]. There is a vast literature on the densest packing of equal discs/squares in a region (e.g. a large circle or square) – see the book [33].

Many geometric packing problems suffer from issues of bit complexity. In particular, there are many packing problems that are not known to lie in NP (e.g., packing discs in a square [11]). This issue is addressed in a recent general approach to geometric approximation [16]. Another direction is to prove that packing problems are complete for the larger class $\exists\mathbb{R}$ (existential theory of the reals) [1].

The distant representatives problem is closely related to problems on imprecise points, where each point is only known to lie within some ε-ball, and the worst-case or best-case representative points, under various measures, are considered. Many geometric problems on points (e.g., convex hulls, spanning trees) have been explored under the model of imprecise points [7, 13, 27, 28].

As mentioned above, the distant representatives problem has application to labelling and visualization, specifically it provides a new approach to the problem of labelling (overlapping) rectangular regions or line segments. Most map labelling research is about labelling point features with rectangular labels of a given size, and the objective is to label as many of the points as possible [18]. There is a small body of literature on labelling line features [12, 36], and even less on labelling regions, except by assuming a finite pre-specified set of label positions [34].

Definitions and Preliminaries

Suppose we are given a set $\mathcal{R}$ of n axis-aligned rectangles in 2D. In the *distant representatives problem*, the goal is to choose a point $p(R) \in R$ for each rectangle R in $\mathcal{R}$ so as to maximize the minimum pairwise distance between points, i.e., we want to maximize $\min_{R,R' \in \mathcal{R}} d_\ell(p(R), p(R'))$, where d_ℓ is the distance-function of our choice. We consider here $\ell = 1, 2, \infty$, i.e., the L_1-distance, the Euclidean L_2-distance and the L_∞-distance. We write δ^*_ℓ for the maximum such distance for $\ell \in \{1, 2, \infty\}$, and omit "$\ell$" when it is clear from the context.

In the *decision version* of the distant representatives problem, we are given not only the rectangles but also a value δ, and we ask whether there exists a set of representative points that have pairwise distances at least δ.

2 Approximating the decision problem

In this section we give an algorithm that takes as input a set $\mathcal{R}$ of axis-aligned rectangles, and a value δ and finds a set of representative points of distance at least δ apart so long as δ is at most some fraction of the optimum, δ^*, for this instance. Let $n = |\mathcal{R}|$ and suppose that the coordinates of the rectangle corners are even integers in the range $[0, D]$ (which guarantees that the rectangle centres also have integer coordinates).

The idea of the algorithm is to overlay a grid of *blocker-shapes* on top of the rectangles as shown in Figure 1, while ensuring that any two blocker-shapes are distance at least δ apart. The hope is to use a matching algorithm to match every rectangle to a unique intersecting blocker-shape. Then, if rectangle R is matched to blocker-shape B, we choose any point in $R \cap B$ as the representative point for B, which guarantees distance at least δ between representative points since the blocker-shapes are distance at least δ apart. The flaw in this plan is that there may be *small* rectangles that do not intersect a blocker shape. To remedy this, we represent a small rectangle by its centre point, and we eliminate any nearby blocker-shapes before running the matching algorithm.

For the L_1 and L_∞ norms we assume that δ is given as a rational number with at most t digits in the numerator and denominator. Because we are using the word-RAM model where we cannot compute square roots, we will work with δ^2 for the L_2 norm. Thus, for the L_2 norm, we assume that we are given δ^2 as a rational number with at most t digits in the numerator and denominator. The bit size of the input is $\Theta(n \log D + t)$. Similarly, for L_2, any output representative point (x, y) will be given as (x^2, y^2). With these nuances of input and output, we express the main result of this section as follows:

▶ **Theorem 1.** *There exists an algorithm* Placement(δ) *that, given input $\ell \in \{1, 2, \infty\}$, rectangles $\mathcal{R}$, and $\delta > 0$, either finds an assignment of representative points for $\mathcal{R}$ of L_ℓ-distance at least δ, or determines that $\delta > \delta^*_\ell / f_\ell$. Here $f_1 = 5, f_2 = \sqrt{34} \approx 5.83, f_\infty = 6$.*

The run-time of the algorithm is $O(n^2 \log n)$ in the word RAM model, i.e., assuming we can do basic arithmetic on numbers of size $O(\log D + t)$ in constant time.

To describe our algorithm, we think of overlaying the $D \times D$ bounding box of the rectangles with a grid of horizontal and vertical lines such that the diagonal distance across a square of the grid is δ. This means that grid lines are spaced γ_ℓ apart, where $\gamma_1 = \delta/2$, $\gamma_2 = \delta/\sqrt{2}$, and $\gamma_\infty = \delta$. For L_2 we will work with $\gamma_2^2 = \delta^2/2$ which is rational. Note that the algorithm does not explicitly construct the grid. Number the grid lines from left to right and bottom to top, and identify a grid point by its two indices. Note that the number of indices is D/γ_ℓ, so the size of each index is $O(\log D + t)$. We imagine filling the grid with *blocker-shapes*, where the chosen shape depends on the norm L_ℓ that is used – see Figure 1.

- For $\ell = 1, 2$, we use *$+$-shapes*. Each $+$-shape consists of the four incident grid-segments of one *anchor* grid-point, where (i, j) is the anchor of a $+$-shape iff i is even and $i \equiv j \mod 4$.
- For $\ell = \infty$, we use *L-shapes*. Each L-shape consists of the two incident grid-segments above and to the right of one *anchor* grid-point, where (i, j) is the anchor of an L-shape iff $i \equiv j \mod 3$.

Observe that, by our choice of grid size γ_ℓ, any two blocker shapes are distance δ or more apart in the relevant norm.

Algorithm Placement(δ)

We now give the rough outline of our algorithm to compute a representative point $p(R)$ for each rectangle R. The details of how to implement each step are given later on. Our algorithm consists of the following steps:

1. Partition the input rectangles into *small* and *big* rectangles. Roughly speaking, a rectangle is *big* if it intersects a blocker-shape, but we give a more precise definition below to deal with intersections on the boundary of the rectangle.
2. For any small rectangle r, let $p(r)$ be the centre of r, i.e., the point where the two diagonals of r intersect each other.
3. If two points $p(r), p(r')$ of two small rectangles r, r' have L_ℓ-distance less than δ, then declare that $\delta^* < f_\ell \delta$, and halt.

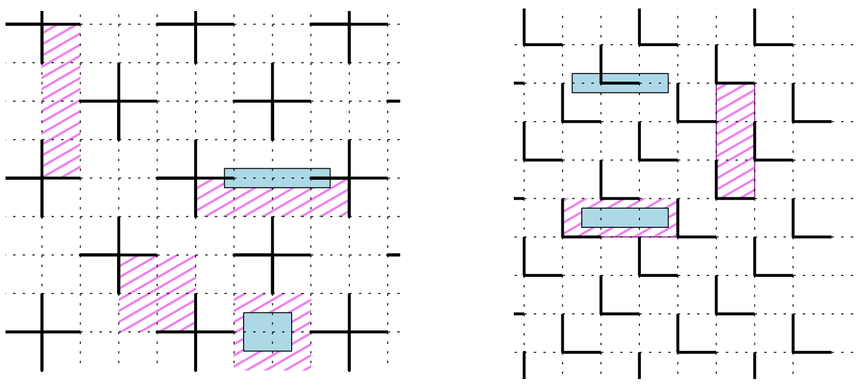


Figure 1 Grids and blocker-shapes. We indicate in each a big and a small rectangle (shaded blue) and some cavities (hatched pink). Small rectangles are contained in cavities. (Left) The grid of $+$-shapes for the L_1 and L_2 norms, with two long cavities and two square cavities. (Right) The grid of L-shapes for the L_∞-norm with two long cavities.

4. Find all the blocker-shapes that are *owned* by small rectangles, where a blocker-shape B *is owned by* a small rectangle r if $p(r)$ has distance strictly less than δ to some point of B. For L_2 we will enlarge ownership as follows: B *is owned by* r if $d_1(p(r), B) < \sqrt{2}\delta$. To justify that this enlarges ownership, note that $d_1(p, q) \le \sqrt{2}d_2(p, q)$ so $d_2(p(r), B) < \delta$ implies $d_1(p(r), B) < \sqrt{2}\delta$.
5. Define a bipartite graph H as follows. On one side, H has a vertex for each big rectangle, and on the other side, it has a vertex for each blocker-shape that is not owned by a small rectangle. Add an edge whenever the rectangle intersects the blocker-shape.
6. Construct a subgraph H^- of H as follows. For any big rectangle, if it has degree more than n in H, then arbitrarily delete incident edges until it has degree n. Also delete any blocker-shape that has no incident edges.
7. Compute a maximum matching M in H^-. We say that it *covers all big rectangles* if every big rectangle has an incident matching-edge in M.
8. If M does not cover all big rectangles, then declare that $\delta^* < f_\ell\delta$ and halt.
9. For each big rectangle R let B be the blocker-shape that R is matched to, and let $p(R)$ be an arbitrary point in $B \cap R$. (This exists since (B, R) was an edge.)
10. Return the set $\{p(R)\}$ (for both big and small rectangles R) as an approximate set of distant representatives.

We now define *big* rectangles more precisely. The intuition is that a rectangle is big if it intersects a blocker shape even if δ is decreased by an infinitesimal amount. Note that, as δ decreases, the blocker-shapes change position and size continuously. More formally, a rectangle is *big* with respect to δ if there is some $\varepsilon_0 > 0$ such that for all ε, $0 \le \varepsilon < \varepsilon_0$, there is a point in the (closed) rectangle and in a blocker-shape (for the blocker-shapes at $\delta - \varepsilon$). The reason for this definition is so the set of big rectangles remains the same if δ is decreased by an infinitesimal amount, a property that becomes relevant when we use the PLACEMENT algorithm to approximately solve the optimization version of distant representatives.

For implementation details and the correctness proof, we need one more definition. A *cavity* is a closed maximal axis-aligned rectangular region with no points of blocker shapes in its interior. We distinguish a *square cavity*, which is a 2×2 block of grid squares (only possible for $+$-shapes), and a *long cavity* which lies between two consecutive grid lines. For $+$-shapes a long cavity is a 1×4 or 4×1 block of grid squares, and for L-shapes, a long cavity is a 1×3 or 3×1 block of grid squares. Observe that any small rectangle is contained in a cavity.

Implementation and Runtime. In order to implement the algorithm efficiently we discuss:

- How to test whether a rectangle is big/small.
- How to find the blocker shapes owned by a small rectangle.
- How to construct H^-.
- How to efficiently compute the matching.

We first show how to find which grid square contains a given point. Identify a grid square by the indices of its lower left grid point. Given a point (x, y) in the plane (e.g., a corner of an input rectangle) the vertical grid line just before x has index i where $i\gamma_\ell \le x < (i+1)\gamma_\ell$ so $i = \lfloor x/\gamma_\ell \rfloor$. For $\ell = 1$, $i = \lfloor 2x/\delta \rfloor$. For $\ell = \infty$, $i = \lfloor x/\delta \rfloor$. For $\ell = 2$, i is the largest natural number such that $i^2 \le 2x^2/\delta^2$, i.e., i is the integer square root of $\lfloor 2x^2/\delta^2 \rfloor$. The integer square root of a number with $O(\log D + t)$ bits can be found in time $O(\log D + t)$ on a word RAM.

We apply the above procedure $O(n)$ times to find the grid squares of all the rectangles' corners and centres. Using the differences and parities of the indices of the grid squares containing the corners, we can test if a rectangle contains points of blocker shapes in its interior or on its boundary. From this, we can test if a rectangle is big or small in constant time. (Note that our complicated rule is really just testing boundary conditions.)

Each small rectangle r owns a constant number of blocker shapes and these can be found by testing a constant number of grid squares that are near $p(r)$.

Next we show how to construct the bipartite graph H^- and compute a maximum matching. Note that blocker-shapes, which form one vertex set of H^-, are specified using $O(\log D + t)$ bits each, although we do not write that in our run-time bounds. To construct H^- we first build a dictionary for the $O(n)$ blocker-shapes owned by small rectangles. Then for each big rectangle R, enumerate blocker-shapes intersecting R in arbitrary order until we have found n that are not owned by a small rectangle, or until we have found all of them, whichever happens first. The run-time for this step is $O(n^2 \log n)$ which will in fact be the bottleneck in our runtime. The graph H^- has $O(n^2)$ vertices and edges.

To find the maximum matching in H^-, we can use the standard algorithm by Hopcroft and Karp [24] which has run-time $O(\sqrt{\nu}|E|)$, where ν is the size of the maximum matching [31, Theorem 16.5]. We have $\nu \le n$ and $|E| = O(n^2)$, so the run-time to find the matching is $O(n^{2.5})$. With appropriate further data structures the runtime of computing the matching can be reduced to $O(n\sqrt{n}\log n)$; see the full version. Therefore the runtime becomes $O(n^2 \log n)$.

Correctness

The algorithm outputs either a set of points or a declaration that $\delta^*_\ell < f_\ell \delta$. We first show that the algorithm is correct if it outputs a set of points.

▶ **Lemma 2.** *If the algorithm returns a point-set, then the L_ℓ-distance between any two points chosen by the algorithm is at least δ.*

Proof. For two small rectangles r, r', this holds since we test $d_\ell(p(r), p(r'))$ explicitly. For any two big rectangles R, R', the two assigned points $p(R)$ and $p(R')$ lie on different blocker-shapes, and hence have distance at least δ. For any big rectangle R and small rectangle r, point $p(R)$ lies on a blocker-shape that is not owned by r, so the blocker shape, and hence $p(R)$, has distance at least δ from $p(r)$. ◀

If the algorithm does not output a set of points, then it outputs a declaration that δ is too large compared to the optimum δ^*, viz., $\delta^*_\ell < f_\ell \delta$. This declaration is made either in Step 3 because the points chosen for small rectangles are too close, or in Step 8 because no matching is found. We must prove correctness in each case, Lemma 3 for Step 3, and Lemma 4 for Step 8. In the remainder of this section we let $p^*(R)$, $R \in \mathcal{R}$ denote an optimum set of distant representatives, i.e., $p^*(R)$ is a point in R and every two such points have L_ℓ-distance at least δ^*_ℓ.

► **Lemma 3.** *If two points $p(r), p(r')$ of two small rectangles r, r' have distance less than δ, then $\delta^*_\ell < f_\ell \delta$.*

Proof. We first show that for any small rectangle r, points $p^*(r)$ and $p(r)$ are close together, specifically, $d_\ell(p^*(r), p(r)) \leq 2.5\gamma_\ell$. Because L_1-distance dominates L_2 and L_∞-distances, it suffices to prove that $d_1(p^*(r), p(r)) \leq 2.5\gamma_\ell$. Any small rectangle is contained in a cavity. The L_1 diameter of a cavity (i.e., the maximum distance between any two points in the cavity) is at most $5\gamma_\ell$ – it is $5\gamma_\ell$ for a long cavity with +-shapes; $4\gamma_\ell$ for a square cavity with +-shapes; and $4\gamma_\ell$ for a long cavity with L-shapes. This implies that any point of r is within distance $2.5\gamma_\ell$ from $p(r)$, the centre of rectangle r.

Now consider two small rectangles r and r' with $d_\ell(p(r), p(r')) < \delta$. We will bound the distance between $p^*(r)$ and $p^*(r')$ by applying the triangle inequality:

$$d_\ell(p^*(r), p^*(r')) \leq d_\ell(p^*(r), p(r)) + d_\ell(p(r), p(r')) + d_\ell(p(r'), p^*(r')) < 2.5\gamma_\ell + \delta + 2.5\gamma_\ell = \delta + 5\gamma_\ell.$$

Plugging in the values $\gamma_1 = \delta/2$, $\gamma_2 = \delta/\sqrt{2}$, and $\gamma_\infty = \delta$, we obtain bounds of 3.5δ, $(1 + 5/\sqrt{2})\delta \approx 4.5\delta$, and 6δ, respectively. Since $f_1 = 5$, $f_2 \approx 5.8$, and $f_\infty = 6$, these bounds are at most $f_\ell\delta$ in all three cases. Thus $\delta^* < f_\ell\delta$, as required. ◄

► **Lemma 4.** *If there is no matching M in H^- that covers all big rectangles, then $\delta^*_\ell < f_\ell\delta$.*

Proof. We prove the contrapositive, using the following plan. Take an optimal set of distant representatives, $p^*(R)$, $R \in \mathcal{R}$ with L_ℓ-distance $\delta^*_\ell \geq f_\ell\delta$. For any big rectangle R, we "round" $p^*(R)$ to a point $b(R)$ that is in R and on a blocker-shape $B(R)$. More precisely, we define $b(R)$ to be a point that is in R, on a blocker-shape, and closest (in L_ℓ distance) to $p^*(R)$. In case of ties, choose $b(R)$ so that the smallest rectangle containing $p^*(R)$ and $b(R)$ is minimal (this is only relevant in L_∞). Break further ties arbitrarily. Observe that $b(R)$ exists, since a big rectangle contains blocker-shape points. Define $B(R)$ to be the blocker-shape containing $b(R)$.

By Lemma 5 (stated below) the pairs $R, B(R)$ form a matching in H that covers all big rectangles. We convert this to a matching in H^- by repeatedly applying the following exchange step. If big rectangle R is matched to a blocker shape $B(R)$ that is not in H^-, then R has degree exactly n in H^-. Not all its n neighbours can be used in the current matching since there are at most $n-1$ big rectangles other than R. So change the matching-edge at R to go to one of the unmatched neighbours in H^- instead. ◄

► **Lemma 5.** *Let R be a big rectangle and let $B = B(R)$. If $\delta^* \geq f_\ell\delta$ then (1) no other big rectangle R' has $B(R') = B$, and (2) no small rectangle owns B.*

The general idea to prove this lemma is to show that either type of collision ($B(R) = B(R')$ or $B(R)$ owned by r) gives points p^* that are "close together", where close together means in a ball of appropriate radius centred at the anchor of $B(R)$.

Let $C_\ell(B)$ be the open L_ℓ-ball centred at the anchor of B and with diameter $f_\ell\delta$. See Figure 2 and note that the diameter of the ball in the appropriate metric is:

$$f_1\delta = 5\delta = 10(\delta/2) = 10\gamma_1 \qquad f_2\delta = \sqrt{34}\delta = \sqrt{68}(\delta/\sqrt{2}) = \sqrt{68}\gamma_2 \qquad f_\infty\delta = 6\delta = 6\gamma_\infty$$

We need a few claims localizing $p^*(R)$ relative to $b(R)$:

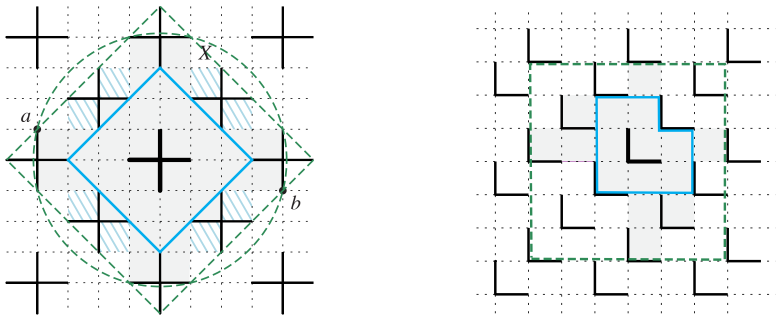


Figure 2 A blocker shape B (heavy black) and $C_\ell(B)$, the L_ℓ-ball of diameter $f_\ell\delta$ centred at B's anchor (dashed green) which is a diamond for L_1 (left), a circle for L_2 (left), and a square for L_∞ (right). The long cavities that touch B are shaded gray. If a small rectangle r owns B, then $p(r)$ lies in C' (in cyan), and r is contained in the union of the gray and blue-hatched regions.

▷ Claim 6. For any big rectangle R, the points $b(R)$ and $p^*(R)$ lie in one long cavity.

Proof. Let T be the rectangle with corners $p^*(R)$ and $b(R)$. By definition of $b(R)$, there are no points of blocker-shapes in or on the boundary of T except $b(R)$. Thus T is contained in a cavity. Furthermore, if T is contained in a square cavity, then we claim that T does not contain the central grid point of the square cavity in its interior (otherwise $b(R)$ could not be the unique point of a blocker shape in T, see Figure 1). Thus T is contained in a long cavity. ◁

▷ Claim 7. Let B be a blocker shape. Let R be a big rectangle with $B(R) = B$. Then $p^*(R)$ is contained in the ball $C_\ell(B)$.

Proof. By the previous lemma, $p^*(R)$ lies in a long cavity that contains a point of B. From Figure 2 we see that any long cavity that contains a point of B lies inside the closed ball $C_\ell(B)$. Furthermore, note that if $p^*(R)$ lies on the boundary of the ball, then it lies on a different blocker shape, contrary to $B(R) = B$. Thus $p^*(R)$ is contained in $R_\ell(B)$. ◁

▷ Claim 8. Let B be a blocker shape. Let r be a small rectangle that owns B. Then $p^*(r)$ is contained in the ball $C_\ell(B)$.

Proof. Recall that $p(r)$ is the centre of r, and that, by the definition of ownership, for $\ell = 1, \infty$ we have $d_\ell(p(r), B) < \delta$, and for $\ell = 2$ we have $d_1(p(r), B) < \sqrt{2}\delta$. Such points $p(r)$ lie in the open region C' drawn in cyan in Figure 2. Here C' is the Minkowski sum of an open ball with B where we use an L_1-ball of radius δ for $\ell = 1$, an L_1-ball of radius $\sqrt{2}\delta$ for $\ell = 2$ and an L_∞-ball of radius δ for $\ell = \infty$.

We next show that r is contained in the ball $C_\ell(B)$. For $\ell = \infty$, r must lie in a long cavity intersecting C', i.e. in the open shaded gray region, thus in $C_\infty(B)$.

For $\ell = 1, 2$, r is contained in a cavity that intersects C'. The union of the cavities that intersect C' consists of the grey and blue-hatched region plus the grid square X and its symmetric counterparts. But observe that a small rectangle that contains points of X has a centre outside C'. Therefore r is contained in $C_\ell(B)$. ◁

Proof of Lemma 5. Let R be a big rectangle and let $B = B(R)$. By Claim 7, $p^*(R)$ lies in $C_\ell(B)$. If there is another big rectangle R' with $B(R') = B$, then $p^*(R')$ also lies in $C_\ell(B)$. If there is a small rectangle r that owns B, then by Claim 8, $p^*(r)$ lies in $C_\ell(B)$.

In either case, the distance between $p^*(R)$ and the other p^* point is less than $f_\ell\delta$, since that is the diameter of $C_\ell(B)$. Thus $\delta^* < f_\ell\delta$. ◀

3 Approximating the optimization problem

In this section we use Placement to design approximation algorithms for the optimization version of the distant representatives problem for rectangles:

▶ **Theorem 9.** *There is an f_ℓ-approximation algorithm for the distant representatives problem on rectangles in the L_ℓ-norm, $\ell = 1, 2, \infty$ with run time $O(n^2(\log n)^2)$ for L_∞ and run time $O(n^2 \text{ polylog}(nD))$ for L_1 and L_2. Here (as before) $f_1 = 5, f_2 = \sqrt{34} \approx 5.83, f_\infty = 6$.*

One complicating factor is that Placement is not monotone, i.e., it may happen that Placement fails for a value δ but succeeds for a larger value δ'. We note that Cabello's algorithm [5] behaves the same way. We deal with L_∞ in Section 3.1 and with L_1 and L_2 in Section 3.2.

We need some upper and lower bounds on δ^*. Note that if the input contains two rectangles that are single identical points, then $\delta^* = 0$. Since this is easily detected, we assume from now on that no two input rectangles are single identical points.

▷ Claim 10. We have $1/n \leq \delta^* \leq 2D$.

Proof. The upper bound is obvious. For the lower bound, place a grid of points distance $\frac{1}{n}$ apart. All rectangles with non-zero dimensions will intersect at least $n+1$ points, and single-point rectangles will hit one point. Since no two single-point rectangles are identical, they can be matched to the sole grid point that overlaps the rectangle. The remaining rectangles can be matched trivially. ◁

Note that a solution with distance at least $\frac{1}{n}$ can be found easily, following the steps above.

3.1 Optimization problem for L_∞

For the L_∞ norm we use the following result about the possible optimum values; a proof is in the full version.

▶ **Lemma 11.** *In L_∞, δ^*_∞ takes on one of the $O(n^3)$ values of the form $(t-b)/k$ where $k \in \{1, \dots, n\}$ and t, b are rectangle coordinates.*

Let Δ be the set of $O(n^3)$ values from the lemma. We can compute the set Δ in $O(n^3)$ time and sort it in $O(n^3 \log n)$ time. Say $\Delta = \{d_1, d_2, \dots, d_N\}$ in sorted order, and set $c_i := d_i/f_\infty$. Because of non-monotonicity, we cannot efficiently find the maximum c_i for which Placement succeeds. Instead, we use binary search to find i such that Placement(c_i) succeeds but Placement(c_{i+1}) fails. Therefore $\delta^*_\infty < f_\infty c_{i+1} = d_{i+1}$ which implies that $\delta^*_\infty \leq d_i = f_\infty c_i$ and the representative points found by Placement(c_i) provide an f_∞-approximation.

To initialize the binary search, we first run Placement(c_N), and, if it succeeds, return its computed representative points since they provide an f_∞-approximation of the optimum assignment. Also note that Placement(c_1) must succeed – if it fails then $\delta^*_\infty < f_\infty c_1 = d_1$, which contradicts $\delta^*_\infty \in \Delta$. Thus we begin with the interval $[1, N]$ and perform binary search on within this interval to find a value i such that Placement(c_i) succeeds but Placement(c_{i+1}) fails.

We can get away with sorting just the $O(n^2)$ numerators and performing an implicit binary search, to avoid the cost of generating and sorting all of Δ. Let t be the sorted array of $O(n^2)$ numerators, which takes $O(n^2 \log n)$ time to generate and sort. The denominators are just $[n]$, so there is no need to generate and sort it explicitly. Define the implicit *sorted* matrix a, where $a[r,c] = t[r]/(n-c)$, for $0 \leq r \leq |t|-1, 0 \leq c \leq n-1$. Each entry of a can be computed in $O(1)$ time. Since the matrix is sorted, the matrix selection algorithm of Frederickson and Johnson [19] can be used to get an element of Δ at the requested index in $O(n \log n)$ time. Using selection, one can perform the binary search on Δ implicitly. While accessing elements of Δ takes more time, it is still less than the time to call PLACEMENT on the element accessed. Each iteration of the binary search is dominated by the runtime of PLACEMENT, so the total runtime is $O(n^2(\log n)^2)$. This proves Theorem 9 for L_∞.

3.2 Optimization problem for L_1 and L_2

In this section we give an approximation algorithm for the optimization version of distant representatives for rectangles in the L_1 and L_2 norms. Define a *critical* value to be a right endpoint of an interval where PLACEMENT succeeds. See Figure 3.

We use the following results whose proofs can be found below.

▶ **Lemma 12.** PLACEMENT *succeeds at critical values, i.e., the intervals where* PLACEMENT *succeeds are closed at the right. Furthermore, a critical value provides an f_ℓ-approximation.*

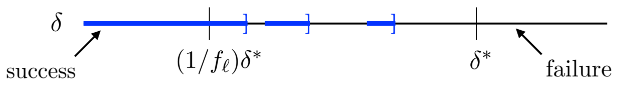


Figure 3 An illustration of critical values.

Thus our problem reduces to finding a critical value.

▶ **Lemma 13.** *The* PLACEMENT *algorithm can be modified to detect critical values.*

▶ **Lemma 14.** *In L_1 any critical value δ is a rational number with numerator and denominator at most $4Dn$. In L_2 for any critical value δ, δ^2 is a rational number with numerator and denominator at most $8D^2n^2$.*

Based on these lemmas, we use continued fractions to find a critical value. We need the following properties of continued fractions.

1. A continued fraction has the form $a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \cdots \frac{1}{a_k}}}$, where the a_i's are natural numbers.
2. Every positive rational number $\frac{a}{b}$ has a continued fraction representation. Furthermore, the number of terms, k, is $O(\log(\max\{a,b\}))$. This follows from the fact that computing the continued fraction representation of $\frac{a}{b}$ exactly parallels the Euclidean algorithm; see [2, Theorem 4.5.2] or the wikipedia page on the Euclidean algorithm [35]. For the same reason, each a_i is bounded by $\max\{a,b\}$.
3. Suppose we don't know $\frac{a}{b}$ explicitly, but we know some bound G such that $a, b \leq G$, and we have a test of whether a partial continued fraction is greater than, less than, or equal to $\frac{a}{b}$. Then we can find the continued fraction representation of $\frac{a}{b}$ as follows. For $i = 1 \dots \log G$, use binary search on $[2..G]$ to find the best value for a_i. Note that the continued fraction with i terms is increasing in a_i for even i and decreasing for odd i, and we adjust the binary search correspondingly. In each step, we have a lower bound and an upper bound on $\frac{a}{b}$ and the step shrinks the interval. If the test runs in time T, then the time to find the continued fraction for $\frac{a}{b}$ is $O(T(\log G)^2)$, plus the cost of doing arithmetic on continued fractions (with no T factor).

Algorithm for L_1, L_2. Run the continued fraction algorithm using Placement (enhanced to detect a critical value) as the test. The only difference from the above description is that we do not have a specific target $\frac{a}{b}$; rather, our interval contains at least one critical value and we search until we find one. At any point we have two values b_l and b_u both represented as continued fractions, where Placement succeeds at b_l and fails at b_u, so there is at least one critical value between them. We can use the initial interval $[1/n, 2D]$. To justify this, note that if Placement$(\frac{1}{n})$ fails , then $f_\ell/n > \delta^*$ by Theorem 1, so we get an f_ℓ-approximation by using the representative points for $\delta = \frac{1}{n}$ (see the remark after Claim 10).

For the runtime, we use the bound $O(T(\log G)^2)$ from point 3 above, plugging in $T = O(n^2 \log n)$ for Placement and the bounds on G from Lemma 14, to obtain a runtime of $O(n^2 \text{polylog}(nD))$, which proves Theorem 9 for L_1 and L_2.

The run-time for Theorem 9 can actually be improved to $O(n^2(\log n)^2)$ (i.e., without the dependence on $\log D$) with an approach that is very specific to the problem at hand (and similar to Cabello's approach). The details are complicated for such a relatively small improvement and hence omitted here.

Missing proofs. For space reasons we can here only give the briefest sketch of the proofs of Lemmas 12, 13, and 14; details are in the full version. A crucial ingredient is to study what must have happened if Placement(δ) goes from success to failure (when viewing its outcome as a function that changes over time δ).

▶ **Observation 15.** *Assume* Placement(δ) *succeeds but* Placement(δ') *fails for some* $\delta' > \delta$*. Then at least one of the following events occurs as we go from* δ *to* δ'*:*

1. *the set of small/big rectangles changes,*
2. *the distance between the centres of two small rectangles equals* $\hat{\delta}$ *for some* $\delta \le \hat{\delta} < \delta'$*,*
3. *the set of blocker-shapes owned by a small rectangle increases,*
4. *the set of blocker-shapes intersecting a big rectangle decreases.*

Roughly speaking, Lemma 12 can now be shown by arguing that such events do not happen in a sufficiently small time-interval before Placement fails (hence the intervals where it fails are open on the left). Lemma 14 holds because there necessarily must have been an event at time δ, and we can analyze the coordinates when events happen. Finally Lemma 13 is achieved by running Placement at time δ and also symbolically at time $\delta + \varepsilon$.

4 Hardness results

In this section we outline NP-hardness and APX-hardness results for the distant representatives problem. For complete details see the full version of the paper. We first show that, even for the special case of unit horizontal segments, the decision version of the problem is NP-complete for L_1 and L_∞ and NP-hard for L_2 (where bit complexity issues prevent us from placing the problem in NP). This L_2 result was proved previously by Roeloffzen in his Master's thesis [30, Section 2.3] but we add details regarding bit complexity that were missing from his proof.

Next, we enhance our reductions to "gap-producing reductions" to obtain lower bounds on the approximation factors that can be achieved in polynomial time. Since our goal is to compare with our approximation algorithms for rectangles, we consider the more general case of horizontal and vertical segments in the plane (not just unit horizontals). Our main result is that, assuming P $\neq$ NP, no polynomial time approximation algorithm achieves a factor better than 1.5 in L_1 and L_∞ and 1.4425 in L_2.

Our reductions are from the NP-complete problem Monotone Rectilinear Planar 3-SAT [10] in which each clause has either three positive literals or three negative literals, each variable is represented by a thin vertical rectangle at x-coordinate 0, each positive [negative] clause is represented by a thin vertical rectangle at a positive [negative, resp.] x-coordinate, and there is a horizontal line segment joining any variable to any clause that contains it. See Figure 4(a) for an example instance of the problem. For n variables and m clauses, the representation can be on an $O(m) \times O(n+m)$ grid.

4.1 NP-hardness

▶ **Theorem 16.** *The decision version of the distant representatives problem for unit horizontal segments in the L_1, L_2 or L_∞ norm is NP-hard.*

Lemma 11 implies that the decision problem lies in NP for the L_∞ norm, even for rectangles. In the full version we show the same for L_1, and we discuss the bit complexity issues that prevent us from placing the decision problem in NP for the L_2 norm.

For our reduction from Monotone Rectilinear Planar 3-SAT we first modify the representation so that each clause rectangle has fixed height and is connected to its three literals via three "wires" – the middle one remains horizontal, the bottom one bends to enter the clause rectangle from the bottom, and the top one bends twice to enter the clause rectangle from the far side. See Figure 4(b). Each wire is directed from the variable to the clause, and represents a literal. The representation is still on an $O(m) \times O(n+m)$ grid.

To complete the reduction to the distant representatives problem we replace the rectangles with variable and clause gadgets constructed from unit horizontal intervals, and also implement the wires using such intervals. The details, which can be found in the full version of the paper, depend on the norm L_ℓ, $\ell = 1, 2, \infty$. We also set a value of δ_ℓ to obtain a decision problem that asks if there is an assignment of a representative point to each interval that is *valid*, i.e., such that no two points are closer than δ_ℓ. We set $\delta_1 = 2$, $\delta_2 = \frac{13}{5}$, and $\delta_\infty = \frac{1}{2}$. An example of the construction for L_1 (with $\delta_1 = 2$) is shown in Figure 4(c).

For the L_2 norm, the bit complexity issue missed in Roeloffzen's reduction [30, Section 2.3] is that the interval endpoints and their distances must have polynomially-bounded bit complexity. We resolve this by using Pythagorean triples (see Figure 5(a)).

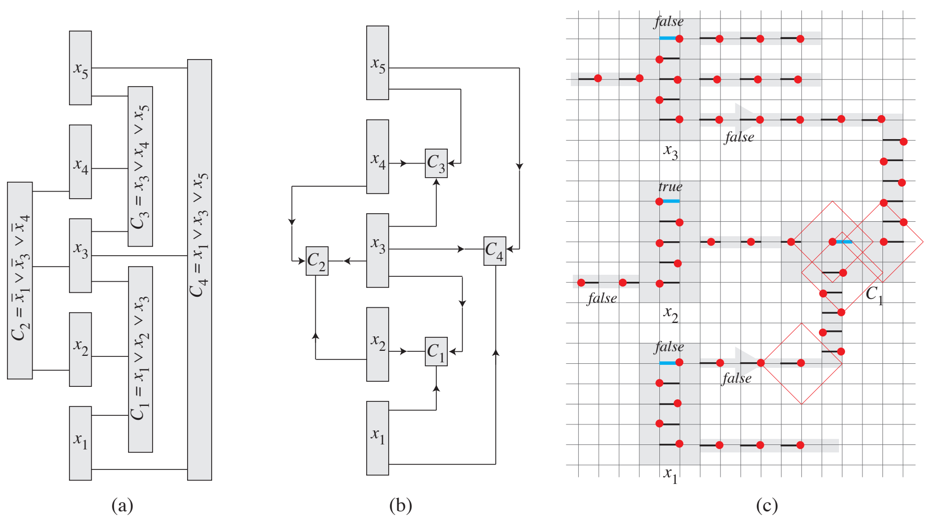
4.2 APX-hardness

In this section, we prove hardness-of-approximation results for the distant representatives problem on horizontal and vertical segments in the plane. Specifically, we prove lower bounds on the approximation factors that can be achieved in polynomial time, assuming P ≠ NP.

▶ **Theorem 17.** *For $\ell = 1, 2, \infty$, let g_ℓ be the constant shown in Table 2. Suppose $P \neq NP$. Then, for the L_ℓ norm, there is no polynomial time algorithm with approximation factor less than g_ℓ for the distant representatives problem for horizontal and vertical segments.*

Table 2 Best approximation ratios that can be achieved unless P=NP.

	L_1	L_2	L_∞
lower bound	$g_1 = 1.5$	$g_2 = 1.4425$	$g_\infty = 1.5$



x_5
x_4
x_3
x_2
x_1
$C_2 = \bar{x}_1 \vee \bar{x}_3 \vee \bar{x}_4$
$C_3 = x_3 \vee x_4 \vee x_5$
$C_1 = x_1 \vee x_2 \vee x_3$
$C_4 = x_1 \vee x_3 \vee x_5$
(a)
C_1
C_2
C_3
C_4
(b)
false
true
C_1
(c)

Figure 4 (a) An instance of Monotone Rectilinear Planar 3-SAT. (b) The modified representation used for our NP-hardness proofs, with wires from variable to clause gadgets. (c) A detail of our NP-hardness construction for clause $C_1 = x_1 \vee x_2 \vee x_3$ in the L_1 norm showing how the truth-value setting x_1 = False, x_2 = True, x_3 = False, permits representative points (shown as red dots) at distance at least $\delta_1 = 2$.

We prove Theorem 17 using a *gap reduction*. This standard approach is based on the fact that if there were polynomial time approximation algorithms with approximation factors better than g_ℓ then the *gap versions* of the problem (as stated below) would be solvable in polynomial time. Thus, proving that the gap versions are NP-hard implies that there are no polynomial time g_ℓ-approximation algorithms unless P=NP.

Recall that δ^*_ℓ is the max over all assignments of representative points, of the min distance between two points.

Gap Distant Representatives Problem.
Input: A set I of horizontal and vertical segments in the plane.
Output:
- YES if $\delta^*_\ell(I) \geq 1$;
- NO if $\delta^*_\ell(I) \leq 1/g_\ell$;
- and it does not matter what the output is for other inputs.

To prove Theorem 17 it therefore suffices to prove:

▶ **Theorem 18.** *The Gap Distant Representatives problem is NP-hard.*

This is proved via a reduction from Monotone Rectilinear Planar 3-SAT, much like in the previous section. The gadgets are simpler because we can use vertical segments, but we must prove stronger properties. Given an instance Φ of Monotone Rectilinear Planar 3-SAT we construct in polynomial time a set of horizontal and vertical segments I such that:

▷ Claim 19. If Φ is satisfiable then $\delta^*_\ell(I) = 1$.

▷ Claim 20. If Φ is not satisfiable then $\delta^*_\ell(I) \leq 1/g_\ell$.

Thus a polynomial time algorithm for the Gap Distant Representatives problem yields a polynomial time algorithm for Monotone Rectilinear Planar 3-SAT. We give some of the reduction details, but defer the proofs of the claims to the full version.

Reduction details

We reduce directly from Monotone Rectilinear Planar 3-SAT.

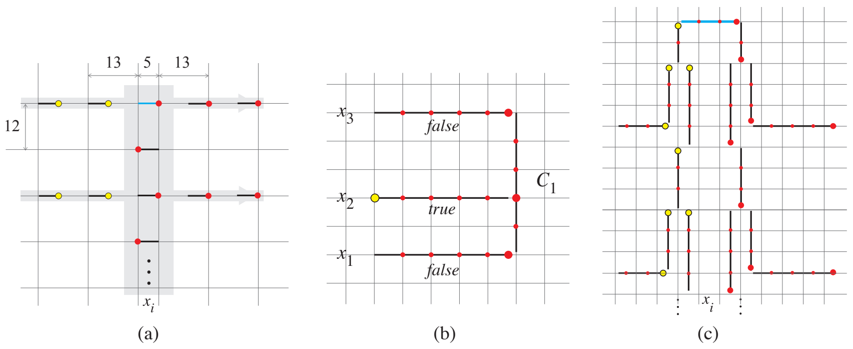


Figure 5 (a) A variable gadget for NP-hardness for L_2, based on Pythagorean triple $5, 12, 13$. To achieve $\delta = 13$ the representative point for the variable interval (in cyan) is forced to the left (true) or the right (false) in which case intervals on the right are also forced. (b) A clause gadget for the APX-hardness reduction, with three horizontal wires attached. For clarity, segments are not drawn all the way to their endpoints. Wires x_1 and x_2 are in the false setting and wire x_2 is in the true setting, which allows the representative point for C_1 to be placed where the x_2 wire meets it, while keeping representative points at least distance 1 apart. (c) The basic splitter gadget for APX-hardness for L_∞ placed on the half grid and showing two wires extending left and two right. The variable segment (in thick cyan) for the variable x_i has its representative point (the large red dot) at the right, which is the false setting. The representative points shown by large red/yellow dots are distance at least 1 apart in L_∞.

Wire. A wire is a long horizontal segment with 0-length segments at unit distances along it, except at its left and right endpoints. See Figure 5(b). The representative point for a 0-length segment must be the single point in the segment (shown as small red dots in the figure). As before, a wire is directed from the variable gadget to the clause gadget. We distinguish a “false setting” where the wire has its representative point within distance 1 of its forward end (at the clause gadget) and a “true setting” where the wire has its representative point within distance 1 of its tail end (at the variable gadget).

Clause gadget. A clause gadget is a vertical segment. Three wires corresponding to the three literals in the clause meet the vertical segment as shown in Figure 5(b). There are 0-length segments at unit distances along the clause interval except where the three wires meet it.

Variable gadget. A variable segment has length 3, with two 0-length segments placed 1 and 2 units from the endpoints. A representative point in the right half corresponds to a false value for the variable, and a representative point in the left half corresponds to a true value. In order to transmit the variable's value to all the connecting horizontal wires we build a “splitter” gadget. The basic splitter gadget for L_∞ is shown in Figure 5(c). The same splitter gadget works for the other norms but we can improve the lower bounds using modified splitter gadgets as described in the full version.

5 Conclusions

We gave good approximation algorithms for the distant representatives problem for rectangles in the plane using a new technique of "imprecise discretization" where we limit the choice of representative points not to a discrete set but to a set of one-dimensional "shapes". This technique may be more widely applicable, and can easily be tailored, for example by using a weighted matching algorithm to prefer representative points near the centres of rectangles.

We also gave the first explicit lower bounds on approximation factors that can be achieved in polynomial time for distant representatives problems.

Besides the obvious questions of improving the approximation factors, the run-times, or the lower bounds, we mention several other avenues for further research.

1. Is the distant representatives problem for rectangles in L_2 hard for existential theory of the reals? Recently, some packing problems have been proved $\exists\mathbb{R}$-complete [1], but they seem substantially harder.
2. Is there a good [approximation] algorithm for any version of distant representatives for a *lexicographic* objective function. For example, suppose we wish to maximize the smallest distance between points, and, subject to that, maximize the second smallest distance, and so on. Or suppose we ask to lexicographically maximize the sorted vector consisting of the n distances from each chosen point to its nearest neighbour. For the case of *ordered* line segments in 1D there is a linear time algorithm to lexicographically minimize the sorted vector of distances between successive pairs of points [4]. It is an open problem to extend this to unordered line segments.
3. What about weighted versions of distant representatives? Here each rectangle R has a weight $w(R)$, and rather than packing disjoint balls of radius δ we pack disjoint balls of radius $w(R)\delta$ centred at a representative point $p(R)$ in R. Again, there is a solution for ordered line segments in 1D [4].

References

1 Mikkel Abrahamsen, Tillmann Miltzow, and Nadja Seiferth. Framework for $\exists\mathbb{R}$-completeness of two-dimensional packing problems. *arXiv preprint arXiv:2004.07558*, 2020. URL: `https://arXiv.org/abs/2004.07558`.

2 Eric Bach and Jeffrey Shallit. *Algorithmic Number Theory: Efficient Algorithms*, volume 1. MIT press, 1996.

3 Christoph Baur and Sándor P. Fekete. Approximation of geometric dispersion problems. *Algorithmica*, 30(3):451–470, 2001. `doi:10.1007/s00453-001-0022-x`.

4 Therese Biedl, Anna Lubiw, Anurag Murty Naredla, Peter Dominik Ralbovsky, and Graeme Stroud. Dispersion for intervals: A geometric approach. In *Symposium on Simplicity in Algorithms (SOSA)*, pages 37–44. SIAM, 2021. `doi:10.1137/1.9781611976496.4`.

5 Sergio Cabello. Approximation algorithms for spreading points. *Journal of Algorithms*, 62(2):49–73, 2007. `doi:10.1016/j.jalgor.2004.06.009`.

6 Jean Cardinal. Computational geometry column 62. *ACM SIGACT News*, 46(4):69–78, 2015. `doi:10.1145/2852040.2852053`.

7 Erin Chambers, Alejandro Erickson, Sándor P. Fekete, Jonathan Lenchner, Jeff Sember, Venkatesh Srinivasan, Ulrike Stege, Svetlana Stolpner, Christophe Weibel, and Sue Whitesides. Connectivity graphs of uncertainty regions. *Algorithmica*, 78(3):990–1019, 2017. `doi:10.1007/s00453-016-0191-2`.

8 Timothy M. Chan. Polynomial-time approximation schemes for packing and piercing fat objects. *Journal of Algorithms*, 46(2):178–189, 2003.

9 Jing Chen, Bo Li, and Yingkai Li. Efficient approximations for the online dispersion problem. *SIAM Journal on Computing*, 48(2):373–416, 2019. doi:10.1137/17M1131027.

10 Mark de Berg and Amirali Khosravi. Optimal binary space partitions in the plane. In *International Computing and Combinatorics Conference*, pages 216–225. Springer, 2010. doi:10.1007/978-3-642-14031-0_25.

11 Erik D. Demaine, Sándor P. Fekete, and Robert J. Lang. Circle packing for origami design is hard. *arXiv preprint arXiv:1008.1224*, 2010. URL: https://arxiv.org/abs/1008.1224.

12 Srinivas Doddi, Madhav V. Marathe, Andy Mirzaian, Bernard M.E. Moret, and Binhai Zhou. Map labeling and its generalizations. In *Proc. 8th Ann. ACM/SIAM Symp. Discrete Algs.(SODA97)*, pages 148–157. SIAM, 1997.

13 Reza Dorrigiv, Robert Fraser, Meng He, Shahin Kamali, Akitoshi Kawamura, Alejandro López-Ortiz, and Diego Seco. On minimum-and maximum-weight minimum spanning trees with neighborhoods. *Theory of Computing Systems*, 56(1):220–250, 2015. doi:10.1007/s00224-014-9591-3.

14 Adrian Dumitrescu and Minghui Jiang. Dispersion in disks. *Theory of Computing Systems*, 51(2):125–142, 2012. doi:10.1007/s00224-011-9331-x.

15 Adrian Dumitrescu and Minghui Jiang. Systems of distant representatives in Euclidean space. *Journal of Combinatorial Theory, Series A*, 134:36–50, 2015. doi:10.1016/j.jcta.2015.03.006.

16 Jeff Erickson, Ivor van der Hoog, and Tillmann Miltzow. Smoothing the gap between NP and $\exists\mathbb{R}$. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1022–1033. IEEE, 2020.

17 Jiří Fiala, Jan Kratochvíl, and Andrzej Proskurowski. Systems of distant representatives. *Discrete Applied Mathematics*, 145(2):306–316, 2005. doi:10.1016/j.dam.2004.02.018.

18 Michael Formann and Frank Wagner. A packing problem with applications to lettering of maps. In *Proceedings of the Seventh Annual Symposium on Computational Geometry*, pages 281–288, 1991.

19 Greg N. Frederickson and Donald B. Johnson. Generalized selection and ranking: sorted matrices. *SIAM Journal on Computing*, 13(1):14–30, 1984.

20 Michael R. Garey, David S. Johnson, Barbara B. Simons, and Robert Endre Tarjan. Scheduling unit–time tasks with arbitrary release times and deadlines. *SIAM Journal on Computing*, 10(2):256–269, 1981. doi:10.1137/0210018.

21 Philip Hall. On representatives of subsets. *Journal of the London Mathematical Society*, 1(1):26–30, 1935.

22 Mhand Hifi and Rym M'hallah. A literature review on circle and sphere packing problems: Models and methodologies. *Advances in Operations Research*, 2009, 2009. doi:10.1155/2009/150624.

23 Dorit S. Hochbaum and Wolfgang Maass. Approximation schemes for covering and packing problems in image processing and VLSI. *Journal of the ACM (JACM)*, 32(1):130–136, 1985. doi:10.1016/j.orl.2010.07.004.

24 John E. Hopcroft and Richard M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM J. Comput.*, 2(4):225–231, 1973. doi:10.1137/0202019.

25 Joseph Y.T. Leung, Tommy W. Tam, Chin 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. doi:10.1016/0743-7315(90)90019-L.

26 Shimin Li and Haitao Wang. Dispersing points on intervals. *Discrete Applied Mathematics*, 239:106–118, 2018. doi:10.1016/j.dam.2017.12.028.

27 Maarten Löffler and Marc van Kreveld. Largest and smallest convex hulls for imprecise points. *Algorithmica*, 56(2):235, 2010. doi:10.1007/s00453-008-9174-2.

28 Maarten Löffler and Marc van Kreveld. Largest bounding box, smallest diameter, and related problems on imprecise points. *Computational Geometry*, 43(4):419–433, 2010. doi:10.1016/j.comgeo.2009.03.007.

29 Roger J. Marshall. Scaled rectangle diagrams can be used to visualize clinical and epidemiological data. *Journal of Clinical Epidemiology*, 58(10):974–981, 2005. doi:10.1016/j.jclinepi.2005.01.018.

30 M.J.M. Roeloffzen. Finding structures on imprecise points. Master's thesis, TU Eindhoven, 2009. URL: https://www.win.tue.nl/~mroeloff/papers/thesis-roeloffzen2009.pdf.

31 Alexander Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*, volume 24. Springer Science & Business Media, 2003.

32 Barbara Simons. A fast algorithm for single processor scheduling. In *19th Annual Symposium on Foundations of Computer Science*, pages 246–252. IEEE, 1978. doi:10.1109/SFCS.1978.4.

33 Péter Gábor Szabó, Mihaly Csaba Markót, Tibor Csendes, Eckard Specht, Leocadio G. Casado, and Inmaculada García. *New Approaches to Circle Packing in a Square: with Program Codes*, volume 6. Springer Science & Business Media, 2007.

34 Frank Wagner, Alexander Wolff, Vikas Kapoor, and Tycho Strijk. Three rules suffice for good label placement. *Algorithmica*, 30(2):334–349, 2001.

35 Wikipedia contributors. Euclidean algorithm — Wikipedia, the free encyclopedia, 2021. [Online; accessed 28-June-2021]. URL: https://en.wikipedia.org/w/index.php?title=Euclidean_algorithm&oldid=1027503317.

36 Alexander Wolff, Lars Knipping, Marc van Kreveld, Tycho Strijk, and Pankaj K. Agarwal. A simple and efficient algorithm for high-quality line labeling. In Peter Atkinson, editor, *GIS and GeoComputation*. Taylor and Francis, 2000. doi:10.1201/9781482268263.