MALMÖ UNIVERSITY

APPTUS

Proceedings of the 33rd European Workshop on Computational Geometry (EuroCG 2017)

April 5-7

Malmö, Sweden

Table of Contents

Session 2B

Session 3A

Session 3B

Session 4A

Session 4B

Session 6A

Session 6B

Session 7A

Session 7B

Session 8A

Session 8B

Best Laid Plans of Lions and Men

Mikkel Abrahamsen* Jacob Holm* Eva Rotenberg* Christian Wulff-Nilsen*

Abstract

We answer the following question dating back to J.E. Littlewood (1885–1977): Can two lions catch a man in a bounded area with rectifiable lakes? The lions and the man are all assumed to be points moving with at most unit speed. That the lakes are rectifiable means that their boundaries are finitely long. This requirement is to avoid pathological examples where the man survives forever because any path to the lions is infinitely long. We show that the answer to the question is not always "yes" by giving an example of a region R in the plane where the man has a strategy to survive forever. R is a polygonal region with 11 holes and the exterior and interior boundaries are pairwise disjoint, simple polygons. Our construction is the first truly two-dimensional example where the man can survive. We prove tightness in the sense that three lions can catch a man in a region with finitely many polygonal lakes. Next, we consider the following game played on the entire plane instead of a bounded area: There is any finite number of unit speed lions and one fast man who can run with speed $1+\varepsilon$ for some value $\varepsilon > 0$. Can the man always survive? We answer the question in the affirmative for any constant $\varepsilon > 0$.

1 Introduction

'A lion and a man in a closed circular arena have equal maximum speeds. What tactics should the lion employ to be sure of his meal?'[1] These words (including the footnote) introduce the now famous lion and man problem, invented by R. Rado in the late thirties, in Littlewood's Miscellany [12]. It was for a long time believed that in order to avoid the lion, it was optimal for the man to run on the boundary of the arena. A simple argument then shows that the lion could always catch the man by staying on the radius OM defined by the man while approaching him as much as possible. However, A.S. Besicovitch proved in 1952 that the man has a very simple strategy (following which he will approach but never reach the boundary) that enables him to avoid capture forever no matter what the lion does. See [12] for the details. One can prove that two lions are enough to catch the man.

A well-known related discrete game is the *cop and robber game*: Let G be a finite connected undirected graph. Two players called cop C and robber R play a game on G according to the following rules: First C and then R occupy some vertex of G. After that they move alternately along edges of G. The cop C wins if at some point in time C and R are on the same vertex. If the robber R can prevent this situation forever, then R wins. The robber has a winning strategy on many graphs, including all cycles of length at least 4. Therefore, the cop player C can be given a better chance by allowing him, say, k cops $C_1, \ldots, C_k$. At every turn C moves any non-empty subset of $\{C_1, \ldots, C_k\}$. Now, the *cop-number of* G is the minimal number of cops needed for C to win. Aigner and Fromme [1] observes that the cop-number of the dodecahedron graph is at least 3, since if there are only 2 cops, the robber can always move to a vertex not occupied by a cop and not in the neighbourhood of any. Furthermore, they prove that the cop-number of any planar graph is at most 3. Thus, the cop-number of the dodecahedron is exactly 3.

Returning to the lion and man game, Bollobás [4] writes that the following open problem was already mentioned by J.E. Littlewood (1885–1977): Can two lions catch a man in a bounded (planar) area with rectifiable lakes? An informal definition of a rectifiable curve is that it has finite length. We require that the boundaries of the lakes and the exterior boundary are all rectifiable curves to avoid pathological examples where the man survives forever because any path to the lions is inifitely long. Bollobás mentions the same problem in a comment in his edition of Littlewood's Miscellany [12] and in [5]. The problem is also stated by Fokkink et al. [9]. Berarducci and Intrigila [3] prove that the man can survive forever (for some initial positions of the man and lions) if the area is a planar embedding of the dodecahedron graph where each edge is a curve with unit length. The proof is essentially the same as the proof by Aigner and Fromme [1] that the cop-number of the dodecahedron is at least 3: Whenever the man is standing at a vertex, there is one of the neighbouring vertices which has distance more than 1. The man can therefore safely run to that vertex. This, however, is a one-dimensional example. They raise the question whether it is possible to replace the one-dimensional edges by two-dimensional thin lines.

We present a truly two-dimensional region R in the plane where two lions are not enough to ever catch the man. We say that R is truly two-dimensional since R

*University of Copenhagen, Denmark

[1] The curve of pursuit (L running always straight at M) takes infinite time, so the wording has its point.

is a polygonal region with holes and the exterior and interior boundaries are all pairwise disjoint, simple polygons – in particular, they are clearly rectifiable. We were likewise inspired by the dodecahendron in the construction of our example.

Rado and Rado [13] consider the problem where there are many lions and one man, but where the game is played in the entire unbounded plane. They prove that the lions can catch the man if and only if the man starts in the interior of the convex hull of the lions. Inspired by that problem, we ask the following question: What if the lions have maximum speed 1 and the man has maximum speed $1+\varepsilon$ for some $\varepsilon > 0$? We prove that for any constant ε and any finite number of lions, such a fast man can survive forever provided that he does not start at the same point as one of the lions. Fast evaders were also studied in [7, 8, 11] in the man-and-lion setting, and in [2, 10] in the cop-and-robber setting.

1.1 Definitions

We follow the conventions of Bollobás et al. [6]. Let $R \subseteq \mathbb{R}^2$ be a region in the plane on which the lion and man game is to be played, and assume that the lion starts at point l_0 and the man at point m_0. We define a *man path* as a function $m\colon [0,\infty) \longrightarrow R$ satisfying $m(0) = m_0$ and the Lipschitz condition $\|m(s) - m(t)\| \leq V \cdot |s-t|$, where V is the speed of the man. In our case, we either have $V = 1$ or, in the case of a fast man, $V = 1+\varepsilon$ for some small constant $\varepsilon > 0$. Note that it follows from the Lipschitz condition that any man path is continuous. A *lion path* l is defined similarly, but the lions we consider always run with at most unit speed.

Let $\mathcal{L}$ be the set of all lion paths and $\mathcal{M}$ be the set of all man paths. Then a *strategy* for the man is a function $M\colon \mathcal{L} \longrightarrow \mathcal{M}$ such that if $l, l' \in \mathcal{L}$ agree on $[0,t]$, then $M(l)$ and $M(l')$ also agree on $[0,t]$. This last condition is a formal way to describe that the man's position $M(l)(t)$, when he follows strategy M, depends only on the position of the lion at points in time before and including time t, i.e., he is not allowed to act based on the lion's future movements. (By the continuity of any man path, the man's position at time t is in fact determined by the lion's position at all times strictly before time t.) A strategy M for the man is *winning* if for any $l \in \mathcal{L}$ and any $t \in [0,\infty)$, it holds that $M(l)(t) \neq l(t)$. Similarly, a strategy for the lion $L\colon \mathcal{M} \longrightarrow \mathcal{L}$ is winning if for any $m \in \mathcal{M}$, it holds that $L(m)(t) = m(t)$ for some $t \in [0,\infty)$. These definitions are extended to games with more than one lion in the natural way.

We call a man strategy M *locally finite* if it satisfies the following property: if l and l' are any two lion paths that agree on $[0,t]$ for some t then the corresponding man paths $M(l)$ and $M(l')$ agree on $[0,t+\delta]$ for some $\delta > 0$ (we allow that δ depends on $l|_{[0,t]}$). Thus, informally, the man commits to doing something for some positive amount of time dependent only on the situation so far. Bollobás et al. [6] prove that if the man has a locally finite winning strategy, then the lion does not have any winning strategy. The argument easily extends to games with multiple lions. At first sight, it might sound absurd to even consider the possibility that the lion has a winning strategy when the man also does. However, it does not follow from the definition that the existence of a winning strategy for the man implies that the lion does not also have a winning strategy. See the paper by Bollobás et al. [6] for a detailed discussion of this (including descriptions of natural variants of the lion and man game where both players have winning strategies). In each of the problems we describe, the winning strategy of the man is locally finite, so it follows that the lions do not have winning strategies. In fact, the strategies we describe satisfy the much stronger condition that they are *equitemporal*, i.e., there is a constant $\Delta > 0$ such that the man at any point in time $i \cdot \Delta$, for $i = 0, 1, \ldots$, decides where he wants to run until time $(i+1) \cdot \Delta$.

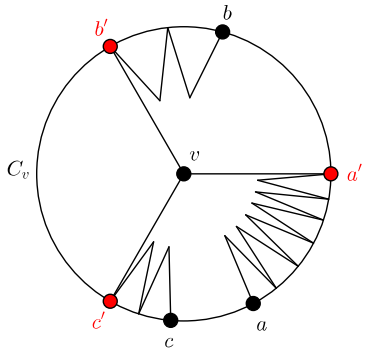


Figure 1: How the planar embedding of the dodecahedron looks in a small disk C_v centered at a vertex v. Regardless of angles between a, b, c, we use the bends to make the three edges meet at v in angles of size $\frac{2\pi}{3}$, and at the same time extend the lengths of the edges suitably, to ensure that all edges are equally long.

1.2 Results

Firstly, we answer the question posed by Littlewood. We construct a polygonal region with holes such that the man has a winning strategy against two lions.

Theorem 1 *There exists a polygonal region R in the plane with 11 polygonal holes where the exterior and interior boundaries are all pairwise disjoint and such that the man has a winning strategy against two lions.*

Secondly, we prove "tightness" in the sense that three lions always suffice to catch a man in a rectifiable region with finitely many rectifiable holes.

Theorem 2 *For any natural number $N \in \mathbb{N}$, for any rectifiable region R in the plane with N rectifiable holes, three lions have a winning strategy against one man.*

Finally, we consider the case where the man is just slightly faster than the lions, in the unbounded plane without obstacles. In this case, the man is able to escape arbitrarily many lions.

Theorem 3 *In the plane $\mathbb{R}^2$, for any $\varepsilon > 0$, a man able to run at speed $1 + \varepsilon$ has a locally finite strategy to escape the convex hull of any number $n \in \mathbb{N}$ of unit-speed lions, provided that the man does not start at the same point as a lion. Thus, the man has a locally finite winning strategy.*

In fact, we prove that the man is able to keep some minimum distance $d_{\varepsilon,n}$ to any lion, where $d_{\varepsilon,n}$ only depends on ε, n, and the initial distances to the lions. Thus, if the n lions and man were disks with radius $< \frac{1}{2}d_{\varepsilon,n}$, the man is still able to escape.

2 Techniques and insights

We will now present a quick overview of the proof strategies for the three theorems.

Sketch of proof, Theorem 1. The idea is to extend a planar embedding of the dodecahedron to a truly two-dimensional region R in the plane by thickening the edges and vertices, such that the man can escape two lions. We thus consider a planar embedding $\mathcal{D}$ of the dodecahedron where all edges have length 4. When thickening the graph $\mathcal{D}$, the vertices are turned into small *vertex areas*, and the edges become *thin strips*. We have to be careful when thickening the edges and vertices. In $\mathcal{D}$, the man has a strategy to survive by only deciding in the vertices where to run next. However, it seems that if we define a specific point in R to correspond to each vertex of $\mathcal{D}$, then if the man only makes decisions in such points, the lions can catch the man. To work around this, we prove:

Claim 1 *In a planar embedding of the dodecahedron where all edges have the same length, if the lions start out sufficiently far away from the man, there exists a winning strategy for the man where he only needs to take the situation into account when he is a quarter of an edge away from the closest vertex.*

Thus, in the region R, the man does not need to decide his path when he is in a vertex area – he can decide beforehand which neighbouring edge he wants to proceed to, so that he ensures always to run through the vertex areas in the shortest possible way.

The planar embedding $\mathcal{D}$ that we consider has the additional property that the three edges meeting at each vertex create angles of size $\frac{2\pi}{3}$. We obtain such a planar embedding from an embedding of the dodecahedron where all edges are straight line segments with length either 1 or 3. Then, in a small circle around each vertex, we extend the edges using some zig-zag bends and at the same time obtain the angle requirement, see Figure 1. We construct a thickening R of $\mathcal{D}$ such that in a disk around each vertex of $\mathcal{D}$, the distances in R between the points where the adjacent edges in $\mathcal{D}$ enter the disk are the same as in $\mathcal{D}$, see Figure 2.

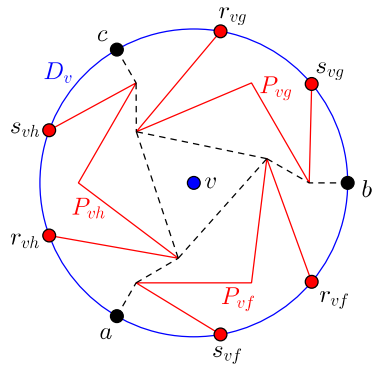


Figure 2: The shortest paths in the circle D_v between any two of a, b, c that avoid crossing the red lines, all have the same length.

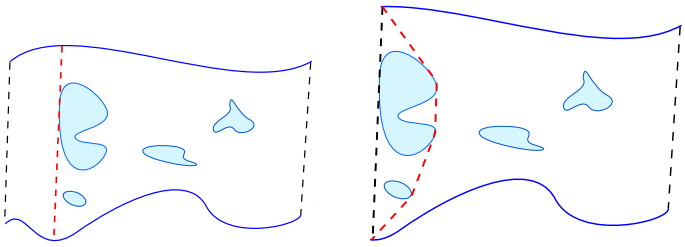


Figure 3: Two lions (black dashed lines) guard a subregion. The third finds a new path (red dashed line) to guard, increasing b or decreasing i.

Let the *quarters* in $\mathcal{D}$ be all points with distance 1 to the closest vertex. We want all quarters in $\mathcal{D}$ to be points in R as well, and we want all pairs of quarters to have the same distances in $\mathcal{D}$ and R. It will then follow from the man's winning strategy in $\mathcal{D}$ that he can also survive in R. We make one lake L_f corresponding to each face f of $\mathcal{D}$. If v is a vertex on f, then the red curve P_{vf} shown in Figure 2 is on the boundary of L_f. Let the vertices on the face f of $\mathcal{D}$ be $uvxyz$, appearing in that counterclockwise order on f. The curves $P_{uf}, P_{vf}, P_{xf}, P_{yf}, P_{zf}$ appear on the boundary of L_f in that order. We connect the end s_{uf} of P_{uf} with the start r_{vf} of P_{vf} – the other curves are connected in a completely analogous way. See Figure 2. There is some edge e_{uv} of $\mathcal{D}$ between u and v which is a polygonal curve. We make a polygonal

curve Q_{uv} corresponding to e_{uv}. Q_{uv} starts at s_{uf} and ends at r_{vf} so that it connects P_{uf} and P_{vf}. Q_{uv} stays near e_{uv} inside f and touches e_{uv} at the corners of e_{uv} which are convex corners of f. It the follows that the shortest paths in R between the quarters of $\mathcal{D}$ all have the same length as in $\mathcal{D}$. Therefore, the man can survive by using the same strategy as he would in $\mathcal{D}$. ◊

Sketch of proof for Theorem 2. The strategy is to let lions guard certain paths in the region, thus restricting the man to a smaller and smaller region. A lion guards a path if she can always reach any point on the path before the man. Whenever two lions restrict the man to an area, the third lion starts guarding a path inside that area, separating the area in at least two smaller areas and making one of the first lions idle. That lion will then start guarding a new path, etc.

We say that a lake is a *boundary lake* if touches a lion-guarded path. The proof goes by induction over the number of lakes and boundary lakes.

Inductively, the man is caught in an area R bounded by four paths (some of which may degenerate to a single point): A lion-guarded path, a lakeside, another lion-guarded path, and another lakeside. Let i and b be the number of lakes and boundary lakes in R, respectively. The third lion starts guarding a path π in R, thus separating R into at least two regions, one of which contains the man and all of which containing $< i$ lakes or $> b$ boundary lakes. That such a path π exists follows nontrivially from a sweeping argument. It then follows that after finitely many iterations, the man is restricted to an area without any lakes. Then the lions change to a much simpler strategy where, in fact, only two lions are needed to catch the man. ◊

Sketch of proof for Theorem 3. We proceed by induction on the number n of lions. We define strategies M_j for the man to keep distance c_j to the first j lions. The j'th strategy yields a curve consisting of line segments all of the same length.

Inductively, the man can keep a safety distance c_{n-1} to the $n-1$ first lions by running at speed $1+\varepsilon_{n-1}$, where $\varepsilon_1 < \varepsilon_2 < \ldots < \varepsilon_n < \varepsilon$. The bends of the curve defined by strategy M_{n-1} are milestones that he runs towards when avoiding n lions. If the n'th lion ℓ_n is in the way, the man makes an *avoidance move*, keeping a much smaller safety distance c_n to ℓ_n and running slightly faster at speed ε_n. Intuitively, when performing avoidance moves, the man runs counterclockwise around a fixed-diameter circle centered at the lion.

After a limited number of avoidance moves, the man can make an *escape move*, where he simply runs towards the milestone defined by the strategy M_{n-1}.

By choosing c_n sufficiently small, we can make sure that the detour caused by the n'th lion is so small that it can only annoy the man once for each of the segments of the strategy M_{n-1}, and thus that he is ensured to have distance at least $c_{i-1}/2$ to the position defined by M_{n-1} and hence not in danger of the $(n-1)$'st lions. ◊

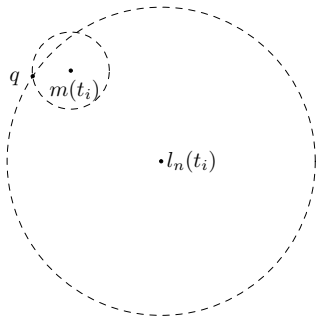


Figure 4: An avoidance move. At time t_j, the man is at $m(t_j)$ and runs to q.

References

[1] M. Aigner and M. Fromme. A game of cops and robbers. *Discrete Applied Mathematics*, 8(1):1–12, 1984.

[2] N. Alon and A. Mehrabian. Chasing a fast robber on planar graphs and random graphs. *Journal of Graph Theory*, 78(2):81–96, 2015.

[3] A. Berarducci and B. Intrigila. On the cop number of a graph. *Advanced Applied Mathematics*, 14(4):389–403, 1993.

[4] B. Bollobás. *The Art of Mathematics: Coffee Time in Memphis*. Cambridge University Press, 2006.

[5] B. Bollobás. The lion and the christian, and other pursuit and evasion games. In D. Schleicher and M. Lackmann, editors, *An Invitation to Mathematics: From Competitions to Research*, pages 181–193. Springer-Verlag Berlin Heidelberg, 2011.

[6] B. Bollobás, I. Leader, and M. Walters. Lion and man—can both win? *Israel Journal of Mathematics*, 189(1):267–286, 2012.

[7] J. Flynn. Lion and man: The boundary constraint. *SIAM Journal on Control*, 11:397–411, 1973.

[8] J. Flynn. Lion and man: The general case. *SIAM Journal on Control*, 12:581–597, 1974.

[9] R. Fokkink, L. Geupel, and K. Kikuta. Open problems on search games. In S. Alpern, R. Fokkink, L. A. Gsieniec, R. Lindelauf, and V. Subrahmanian, editors, *Search Theory: A Game Theoretic Perspective*, chapter 5, pages 181–193. Springer-Verlag New York, 2013.

[10] F. V. Fomin, P. A. Golovach, J. Kratochvíl, N. Nisse, and K. Suchan. Pursuing a fast robber on a graph. *Theoretical Computer Science*, 411:1167–1181, 2010.

[11] J. Lewin. The lion and man problem revisited. *Journal of Optimization Theory and Applications*, 49(3):411–430, 1986.

[12] J. E. Littlewood. *Littlewood's miscellany*. Cambridge University Press, 1986.

[13] P. A. Rado and R. Rado. More about lions and other animals. *Mathematical Spectrum*, 7(3):89–93, 1974/75.

Parallel Motion Planning: Coordinating a Swarm of Labeled Robots with Bounded Stretch*

Erik D. Demaine† Sándor P. Fekete‡ Phillip Keldenich‡ Henk Meijer§ Christian Scheffer‡

Abstract

We present a collection of results for parallel motion planning, in which the objective is to reconfigure a swarm of labeled disk-shaped objects into a given target arrangement. This problem is of significant importance for a wide range of practical challenges, with potential applications to coordinated motion planning of ground robots, self-driving cars, and/or drone swarms, in addition to air traffic control, and human team coordination (e.g., in sports, military, or fire fighting).

We solve an open problem by Overmars dating back to 2006 by designing a constant-factor approximation algorithm for minimizing the execution time of a *parallel motion plan* for a rectangular grid of robots, and a desired permutation of those robots, where, in each round, every robot can move to any neighboring location whose robot is simultaneously leaving to another location. In fact, our algorithm achieves constant *stretch factor*: if all robots ultimately want to move to a location at most d units away, then the computed parallel motion plan requires only $\mathcal{O}(d)$ rounds.

Furthermore, we provide lower and upper bound results for the corresponding continuous and unlabeled versions of the problem setting.

1 Introduction

Since the beginning of computational geometry, robot motion planning and especially multi-robot coordination has received a considerable amount of attention. Even in the groundbreaking work by Schwartz and Sharir [11] from the early 1980s, one of the challenges was coordinating the motion of *several* disk-shaped objects among obstacles. Their algorithms run in time polynomial in the complexity of the obstacles, but exponential in the number of disks; moreover, it was shown by Hopcroft et al. [5] that the reachability of a given target configuration is PSPACE-complete to decide. This illustrates that a major aspect of the complexity arises not just from dealing with obstacles, but from interaction between the individual robots. In addition, a growing number of applications focus solely on robot interaction, even in settings in which obstacles are of minor importance, such as air traffic control or swarm robotics, where the goal is overall efficiency, rather than individual navigation.

With the hardness of multi-robot coordination being well known, there is still a huge demand for positive results with provable performance guarantees. In this paper, we provide significant progress in this direction, with a broad spectrum of results.

1.1 Our Results

For the problem of minimizing the total time needed to reconfigure a system of labeled circular robots in a grid environment, we give an $\mathcal{O}(1)$-approximation, i.e. bounded *stretch*, for optimal parallel motion planning, solving an open problem stated by Overmars [9] in 2006. See Theorem 1.

We extend our approach to establish constant stretch for the generalization of *colored* case, for which unlabeled disks are another special case; see Theorem 2. For the continuous case of N disks and arbitrary density, we establish a lower bound of $\Omega(N^{1/4})$ and an upper bound of $\mathcal{O}(\sqrt{N})$ on the achievable stretch, see Theorem 3 and Theorem 4.

1.2 Related Work

Multi-object motion planning problems have received a tremendous amount of attention from a wide spectrum of areas. Due to limited space, we focus on algorithmic work with a focus on geometry.

In the presence of obstacles, Aronov et al. [2] demonstrate that for up to three robots, a path can be constructed efficiently, if one exists. Schwartz and Sharir [11] consider the case of several disk-shaped objects between polygonal obstacles. They give algorithms for deciding reachability of a given target configuration. The algorithms run in time polynomial in the complexity of the obstacles, but exponential in the number of disks. Hopcroft et al. [5] prove that it is PSPACE-complete to decide reachability of a given target configuration, even when restricted to rectan-

*This work was partially supported by the DFG Reserach Unit Controlling Concurrent Change, funding number FOR 1800, project FE407/17-2, Conflict Resolution and Optimization.

†MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, Massachusetts, USA, edemaine@mit.edu

‡Department of Computer Science, TU Braunschweig, Germany, {s.fekete,p.keldenich,c.scheffer}@tu-bs.de

§Science Department, University College Roosevelt, Middelburg, The Netherlands, h.meijer@ucr.nl

gular objects in a rectangular region. Dumitrescu and Jiang [4] consider minimizing the *number* of moves of a set of disks into a target arrangement without obstacles. They prove that the problem remains NP-hard for congruent disks even when the motion is restricted to sliding.

In both discrete and continuous variants of the problem, the objects can be *labeled*, *colored* or *unlabeled*. In the *colored* case, the objects are partitioned into k groups and each target position can only be covered by an object with the right color. This case was recently considered by Solovey and Halperin [12], who present and evaluate a practical sampling-based algorithm. In the *unlabeled* case, the objects are indistinguishable and each target position can be covered by any object. This scenario was first considered by Kloder and Hutchinson [6], who presented a practical sampling-based algorithm. Turpin et al. [15] prove that it is possible to find a solution in polynomial time, if one exists. This solution is optimal with respect to the longest distance traveled by any one robot. However, their results only hold for disk-shaped robots under additional restrictive assumptions on the free space. For unit disks and simple polygons, Adler et al. [1] provide a polynomial-time algorithm under the additional assumption that the start and target positions have some minimal distance from each other. Under similar distance assumptions, Solovey et al. [14] provide a polynomial-time algorithm that produces a set of paths that is no longer than $\mathrm{OPT} + 4m$, where m is the number of robots. However, they do not consider the makespan, but only the total path length. On the negative side, Solovey and Halperin [13] prove that the unlabeled multiple-object motion planning problem is PSPACE-hard, even when restricted to unit square objects in a polygonal environment.

On grid graphs, approaches for the problem (see Kunde [8] and Cheung and Lau [3]) typically assume that at least a constant number of packets can be held at any processor which means that a constant number of robots may overlap in the context of our problem setting. On the other hand, on grid graphs, the problem resembles the generalization of the 15-puzzle, for which Wagner [16] and Kornhauser et al. [7] give an efficient algorithm that decides reachability of a target configuration and provide both lower and upper bounds on the number of moves required. Ratner and Warmuth [10] prove finding a shortest solution for this puzzle remains NP-hard.

2 Preliminaries

In the grid setting considered in Section 3, robots are arranged in an $n \times m$-rectangle P which is dual to a grid graph $G = (V, E)$. A *configuration* of P is an injective mapping $C : V \to \{1, \ldots, k, \perp\}$, where $\{1, \ldots, k\}$ are the labels of the $k \leqslant |P|$ robots to be moved, and C does not have to be injective with respect to the empty squares denoted by $\perp$. The inverse image of a robot's label ℓ is denoted by $C^{-1}(\ell)$. d is the maximum distance between a robot's start and target position.

A configuration $C_1 : V \to \{1, \ldots, k, \perp\}$ can be *transformed* into another configuration $C_2 : V \to \{1, \ldots, k, \perp\}$, denoted $C_1 \to C_2$, if $C_1^{-1}(\ell) = C_2^{-1}(\ell)$ or $(C_1^{-1}(\ell), C_2^{-1}(\ell)) \in E$ holds for all $\ell \in \{1, \ldots, k\}$, i.e., if each robot does not move or moves to one of the four adjacent squares. Furthermore, two robots cannot exchange their squares in one transformation step. The number of steps in a sequence of transformations is called its *makespan*. Given a start configuration C_s and a target configuration C_t, the *optimal makespan* is the minimum number of steps in a transformation sequence starting with C_s and ending with C_t.

For the continuous setting of Section 4, we consider N robots $R := \{1, \ldots, N\} \subseteq \mathbb{N}$. A *movement* of a robot r is a curve $m_r : [0, T_r] \to \mathbb{R}^2$, such that $||m_r'(t)'|_2 \leqslant 1$ holds for all points in time $t \in [0, T_r]$. Let $m_i : [0, T_i] \to \mathbb{R}^2$ and $m_j : [0, T_j] \to \mathbb{R}^2$ be two movements; m_i and m_j are *compatible* if the corresponding robots do not intersect at any time.A *movement* of R is a set of compatible movements $\{m_1, \ldots, m_N\}$, one for each robot. The *(continuous) makespan* of a movement $\{m_1, \ldots, m_N\}$ is defined as $\max_{r \in R} T_r$. A movement $\{m_1, \ldots, m_N\}$ *realizes* a pair of start and target configurations $\mathcal{S} := (\{s_1, \ldots, s_N\}, \{t_1, \ldots, t_N\})$ if $m_r(0) = s_r$ and $m_r(T_r) = t_r$ hold for all $r \in R$. We are searching for a movement $\{m_1, \ldots, m_N\}$ realizing $\mathcal{S}$ with minimal makespan.

3 Labeled Grid Permutation

Let $n \geqslant m \geqslant 2$, $n \geqslant 3$ and let P be a $n \times m$-rectangle. By filling possibly empty squares with dummy robots, we may assume $k = |P| = nm$.

Our main result is the following:

Theorem 1 *There is an algorithm with runtime $\mathcal{O}(dmn)$ that, given an arbitrary pair of start and target configurations of an $n \times m$-rectangle with maximum distance d between any start and target position, computes a schedule of makespan $\mathcal{O}(d)$, i.e., an approximation algorithm with constant stretch.*

On a high level, our algorithm first computes the maximal Manhattan distance d between a robot's start and target position. Then we partition P into a set T of pairwise disjoint rectangular *tiles*, where each tile $t \in T$ is an $n' \times m'$-rectangle for $n', m' \leqslant 24d$. We then use an algorithm based on flows to guarantee that all robots are in their target tile, see Figure 1. Once all robots are in the correct tile, we use a sorting algorithm, *rotate sort*, for meshes simultaneously on all tiles to move each robot to the correct position within its target tile.

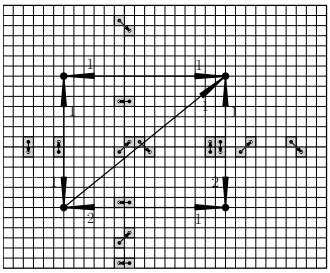


Figure 1: A tiling of an 26×32-rectangle into four tiles with $d = 1$ and the corresponding dual graph. Robots not in their target tile are illustrated by small dots. Their target positions are depicted as white disks.

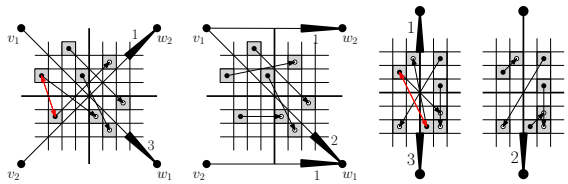
3.1 Outline of the Approximation Algorithm

We model the movements of robots between tiles as a flow f_T, using the weighted directed graph $G_T = (T, E_T, f_T)$, which is dual to the tiling T defined in the previous section. In G_T, we have an edge $(v, w) \in E_T$ if there is at least one robot that has to move from v into w. Furthermore, we define the weight $f_T((v, w))$ of an edge as the number of robots that move from v to w. As P is fully occupied, f_T is a *cyclic flow*, i.e., a flow with no sources or sinks, in which flow conservation has to hold at all vertices. We observe that G_T is a grid graph with additional diagonal edges and thus has degree at most 8. This is due to the fact that the side lengths of the tiles are larger than d as enforced by construction of the tiling.

While the maximum edge value of f_T may be $\Theta(d^2)$, only $\mathcal{O}(d)$ robots can possibly leave a tile within a single transformation step. Therefore, we decompose the flow f_T of robots into a *partition* consisting of $\mathcal{O}(d)$ *subflows*, where each individual robot's motion is modeled by exactly one subflow and each edge in the subflows has value at most d. Each subflow is then *realized* in a single transformation step. To facilitate the decomposition into subflows, we first preprocess G_T. The algorithm consists of the following subroutines:

Step 1: Compute d, the tiling T and the flow G_T, see Figure 1 for the basic idea.

Step 2: Remove intersecting and bidirectional edges from G_T, see the Figure below for the basic idea.



Step 3: Compute a partition of G_T into $\mathcal{O}(d)$ subflows with edge flows upper bounded by d.

Step 4: Realize the $\mathcal{O}(d)$ subflows using $\mathcal{O}(d)$ transformation steps, see Figures 2 and 3 for the basic ideas.

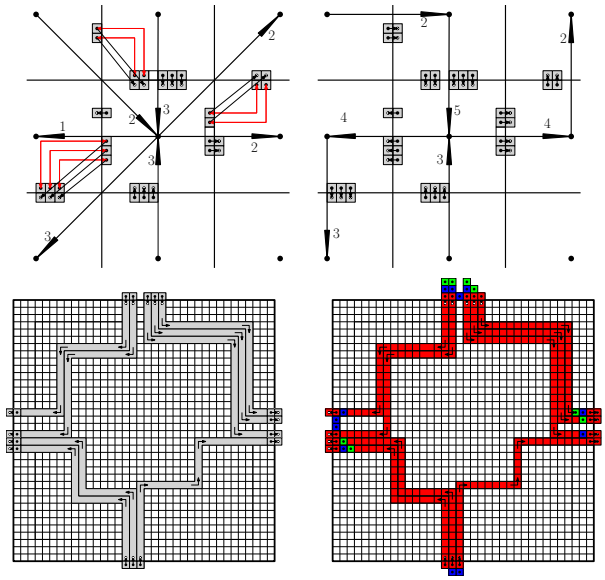


Figure 2: Remove diagonal edges (top) and than apply the main approach (bottom) for realizing a subflow.

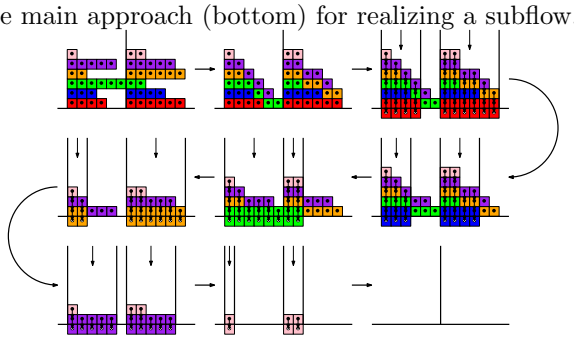


Figure 3: Realizing a sequence of subflows by stacking the rows of robots to be moved onto each other in the order in which the subflows are realized. Each color indicates a separate subflow.

Step 5: Simultaneously apply a sorting algorithm, *rotate sort*, for meshes to all tiles, moving each robot to its target position.

4 Variants on Labeling

A different version is the unlabeled variant, in which all robots are the same. A generalization of both this and the labeled version arises when robots belong to one of k color classes, with robots from the same color class being identical.

Theorem 2 *There is an algorithm with running time $\mathcal{O}(k(mn)^{1.5}\log(mn) + dmn)$ that computes, given start and target images I_s, I_t with maximum distance d between any start and target position, an $\mathcal{O}(1)$-approximation of the optimal makespan M and a corresponding sequence of transformation steps.*

The basic idea is to transform the given labeled problem setting into an unlabeled problem setting by solving a geometric bottleneck problem.

5 Continuous Motion

The continuous geometric case considers N unit disks that have to move into a target configuration in the plane; the velocity of each robot is bounded by 1, and we want to minimize the makespan. For dense arrangements of disks, we can show that constant stretch can *not* be achieved.

Theorem 3 *There is an instance with optimal makespan $M \in \Omega(N^{1/4}) = \Omega(dN^{1/4})$ where $d \in \Theta(1)$, see Figure 4.*

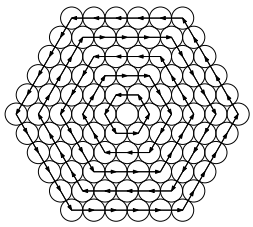


Figure 4: The start and target configurations of our lower-bound construction.

The basic idea of the proof of Theorem 3 is the following. Let $\{m_1, \ldots, m_N\}$ be an arbitrary movement with makespan M. We show that there must be a point in time $t \in [0, M]$ where the area of $\mathrm{Conv}(m_1(t), \ldots, m_N(t))$ is lower-bounded by $cN + \Omega(N^{3/4})$, where cN is the area of $\mathrm{Conv}(m_1(0), \ldots, m_N(0))$. Assume $M \in o\left(N^{1/4}\right)$ and consider the area of $\mathrm{Conv}(m_1(t'), \ldots, m_N(t'))$ at some point $t' \in [0, M]$. This area is at most $cN + \mathcal{O}(\sqrt{N}) \cdot o\left(N^{1/4}\right)$ which is a contradiction.

On the other hand, we can give a non-trivial but non-constant upper bound on the possible stretch.

Theorem 4 *There is an algorithm that computes a movement plan with continuous makespan in $\mathcal{O}(d + \sqrt{N})$. If $d \in \Omega(1)$, this implies a $\mathcal{O}(\sqrt{N})$-approximation algorithm.*

The approach of Theorem 4 applies an underlying grid with mesh size $2\sqrt{2}$. Our algorithm (1) moves the robots to vertices of the grid, (2) applies our $\mathcal{O}(1)$-approximation for the discrete case, and (3) moves the robots from the vertices of the grid to their targets.

References

[1] A. Adler, M. de Berg, D. Halperin, and K. Solovey. Efficient multi-robot motion planning for unlabeled discs in simple polygons. In *Algorithmic Foundations of Robotics XI*, pages 1–17. Springer, 2015.

[2] B. Aronov, M. de Berg, A. F. van der Stappen, P. Švestka, and J. Vleugels. Motion planning for multiple robots. *Discrete & Computational Geometry*, 22(4):505–525, 1999.

[3] S. Cheung and F. C. M. Lau. Mesh permutation routing with locality. *Information Processing Letters*, 43(2):101–105, 1992.

[4] A. Dumitrescu and M. Jiang. On reconfiguration of disks in the plane and related problems. *Computational Geometry: Theory and Applications*, 46:191–202, 2013.

[5] J. E. Hopcroft, J. T. Schwartz, and M. Sharir. On the complexity of motion planning for multiple independent objects; PSPACE-hardness of the warehouseman's problem. *Int. J. Robotics Research*, 3(4):76–88, 1984.

[6] S. Kloder and S. Hutchinson. Path planning for permutation-invariant multi-robot formations. In *IEEE Trans. Robotics*, volume 22, pages 650–665. IEEE, 2006.

[7] D. Kornhauser, G. Miller, and P. Spirakis. Coordinating pebble motion on graphs, the diameter of permutation groups, and applications. In *Annual Symposium on Foundations of Computer Science, 1984*, SFCS '84, pages 241–250, 1984.

[8] M. Kunde. Routing and sorting on mesh-connected arrays. In *VLSI Algorithms and Architectures: 3rd Aegean Workshop on Comp. (AWOC 88)*, pages 423–433. Springer, 1988.

[9] M. Overmars. Contributed open problem. In S. P. Fekete, R. Fleischer, R. Klein, and A. Lopez-Ortiz, editors, *Algorithmic Foundations of Programmable Matter, Dagstuhl Seminar 06421*, 2006. http://www.dagstuhl.de/de/programm/kalender/semhp/?semnr=06421.

[10] D. Ratner and M. K. Warmuth. Finding a shortest solution for the $N \times N$ extension of the 15-puzzle is intractable. In *Proc. AAAI Conf. Artificial Intelligence*, pages 168–172, 1986.

[11] J. T. Schwartz and M. Sharir. On the piano movers' problem: III. Coordinating the motion of several independent bodies: the special case of circular bodies moving amidst polygonal barriers. *Int. J. Robotics Research*, 2(3):46–75, 1983.

[12] K. Solovey and D. Halperin. k-color multi-robot motion planning. *Int. J. Robotics Research*, 33(1):82–97, 2014.

[13] K. Solovey and D. Halperin. On the hardness of unlabeled multi-robot motion planning. In *Robotics: Science and Systems (RSS)*, 2015.

[14] K. Solovey, J. Yu, O. Zamir, and D. Halperin. Motion planning for unlabeled discs with optimality guarantees. In *Robotics: Science and Systems (RSS)*, 2015.

[15] M. Turpin, N. Michael, and V. Kumar. Trajectory planning and assignment in multirobot systems. In *Algorithmic foundations of robotics X*, pages 175–190. Springer, 2013.

[16] R. M. Wilson. Graph puzzles, homotopy, and the alternating group. *Journal of Combinatorial Theory, Series B*, 16(1):86–96, 1974.

Forming Tile Shapes with a Single Robot*

Robert Gmyr† Irina Kostitsyna‡ Fabian Kuhn§ Christian Scheideler† Thim Strothmann †

1 Introduction

We investigate the problem of *shape formation* with *robots* on *tiles* in which a collection of robots has to rearrange a set of movable tiles to form a desired shape. In this preliminary work we consider the case of a single robot operating on an arbitrary number of tiles and present first results towards the formation of simple shapes. Our ultimate goal is to investigate how multiple robots can cooperate to speed up the process of shape formation.

Model. We consider a single *robot* acting on a finite set of *hexagonal tiles*. The tiles are *passive*, i.e., they do not perform any computation and cannot move on their own. The tiles may form any structure so that their centers coincide with nodes of a triangular grid graph, as shown in Figure 1, and there is at most one tile per node.

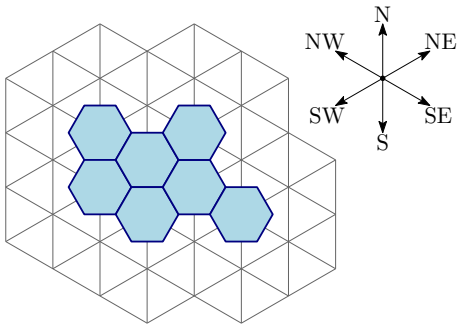


Figure 1: An example tile configuration. The top right part of the figure shows the compass directions we use to describe the movement of the robot.

The robot is *active* and may occupy any node of the grid graph. It is a *deterministic finite automaton* that operates in *look-compute-move* cycles. In the *look* phase the robot can observe the node it occupies and the six neighbors of that node. For each of these nodes it can determine whether there is a tile placed at that node. In the *compute* phase the robot can use this information together with its state to determine its next move and to change its state. In the *move* phase the robot can take a tile from its current node, place a tile it is carrying at that node, or move to an adjacent node while possibly carrying a tile with it. The robot can carry at most one tile.

Note that even though we describe the algorithms as if the robot knew its global orientation, we do not actually require the robot to have a compass. For the algorithms presented in this paper, it is enough for the robot to be able to maintain its relative orientation with respect to its original orientation.

Problem Statement. A *configuration* consists of the positions (i.e., the occupied nodes) of the tiles and the position and state of the robot. We define a configuration to be *connected* if the subgraph induced by the nodes that are occupied by the tiles (including a tile carried by the robot) is connected. In the *triangle formation problem* we are given an arbitrary connected configuration in an infinite grid graph with a robot in the initial state and the goal is to rearrange the tiles into an equilateral triangle with the help of the robot while having a connected configuration at the beginning of every look-compute-move cycle.

We aim at maintaining connectivity of the tile structure. Consider a scenario where a tile structure floats in a liquid. Connected components of a disconnected structure might float apart. Thus, we want our techniques to be applicable to scenarios where it is important to maintain fixed tile positions (relative to each other). To be applicable also to nano-systems, we assume the robot just to have the computational power of a finite automaton.

Note that the requirement for connectivity effectively restricts the movement of the robot but there are no restrictions concerning the grid graph since it is assumed to be infinite in every direction. Also note that if the number of tiles is not a triangular number, one side of the triangle is only partially occupied by tiles.

Related Work. There is a number of approaches to shape formation in the literature that use agents that fall somewhere in the spectrum between passive and active. For example, *tile-based self-assembly* [8] uses passive tiles that bond to each other to form shapes. A variant of *population protocols* proposed in [7] uses agents that are partly passive (i.e., they cannot control their movement) and partly active (i.e., upon meeting another, they can perform a computation and decide whether they want to form a bond). Fi-

*This work was begun at the Dagstuhl Seminar on Algorithmic Foundations of Programmable Matter, July 3–8, 2016.

†University of Paderborn, Germany
‡Université libre de Bruxelles (ULB), Brussels, Belgium
§University of Freiburg, Germany

nally, the *amoebot model* [4], the *nubot model* [11], and the modular robotic model proposed in [6] use agents that are completely active in that they can compute and control their movement. All of these approaches have in common that they consider a *single type* of agent. In contrast, we investigate a model that uses a *combination* of *active* and *passive* agents.

When arguing about a robot that traverses a tile structure without moving tiles, our model essentially reduces to an instance of the ubiquitous *agents on graphs* model. The vast amount of research on this model covers many interesting problems such as Gathering and Rendezvous (e.g. [9]), Intruder Caption and Graph Searching (e.g. [1, 5]), and Graph Exploration (e.g. [2]). Some approaches are also known that allow agents to move tiles (e.g. [3, 10]) but these focus on computational complexity issues or agents that are more powerful than finite automata.

2 A Naive Approach

In a naive approach to shape formation, the robot could iteratively search for a tile that can be removed without disconnecting the tile structure and then move that tile to some position such that the shape under construction is extended. While there always is a tile that can be safely removed, the following theorem shows that, in general, the robot cannot find it, which makes this naive approach infeasible.

Theorem 1 *The robot cannot find a tile that can be removed without disconnecting the tile structure.*

Proof. Suppose that there is an algorithm that allows the robot to find such a tile. Let s be the number of states used by the algorithm. Consider the execution of the algorithm on a hollow hexagon of side length ℓ where the robot is initially placed on a vertex of the hexagon as depicted in the left part of Figure 2. We subdivide the execution into *phases* where we define a new phase to start whenever the robot visits a vertex of the hexagon. Note that the algorithm runs for at most $6s$ phases before the robot chooses the tile because if it would run for more phases the robot would visit the same vertex twice in the same state and therefore the algorithm would enter an infinite loop.

The way the robot traverses the hexagon depends on the side length ℓ. We define the *traversal sequence* associated with ℓ as $((v_1, q_1), (v_2, q_2), \ldots, (v_k, q_k))$ where k is the number of phases the algorithm takes until the tile is chosen, v_i is the vertex occupied by the robot at the beginning of phase i, and q_i is the state of the robot at the beginning of phase i. Since the algorithm takes at most $6s$ phases to choose the tile (independently of ℓ), there are at most $(6s)^{6s}$ distinct traversal sequences. Hence, there is a finite number of traversal sequences and an infinite number of side lengths which implies, according to the pigeonhole principle, that there must be an infinite set of side lengths L that have the same traversal sequence.

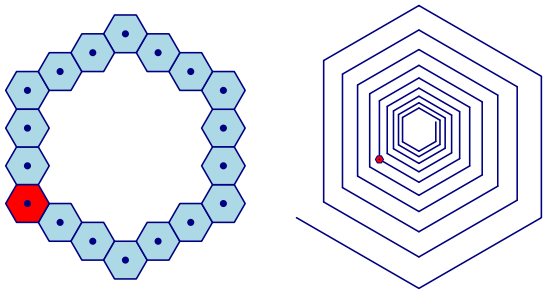


Figure 2: Left: The hollow hexagon of side length $\ell = 4$. Right: An example of the tile structure S. The red mark represents the initial position of the robot.

Based on this observation, we now define a tile structure S for which the algorithm fails to find a tile that can be safely removed. This tile structure essentially consists of a spiral as depicted in the right part of Figure 2. We start at an arbitrary node of the triangular grid graph and construct an outward spiral consisting of $24s$ line segments. The first line segment of the spiral goes north and each following line segment takes a 60° clockwise turn. The lengths of the line segments are chosen from L in such a way that the segments stay separated. This is possible since L is an infinite set and therefore we can always choose sufficiently large segment lengths. We initially place the robot at the end of the $12s$-th line segment.

It remains to show that the algorithm fails to find a tile that can be safely removed when being executed on S. As above, we subdivide the execution of the algorithm into phases where we define a new phase to start whenever the robot visits a vertex of the spiral (i.e., a tile where two line segments meet). It is easy to show using induction on the phases that the robot traverses S in a way that corresponds to the traversal sequence associated with the side lengths in L. Consequently, the robot chooses a tile that is neither the start tile nor the end tile of the spiral. Since these two tiles are the only tiles that can be safely removed from S, the algorithm fails. This contradicts the assumption that the algorithm works correctly and therefore shows that there is no such algorithm. □

3 Taking a Detour via a Line

We now present an approach that avoids the pitfall of the naive approach by first rearranging the tiles into a straight line. Whenever a tile that cannot be removed without disconnecting the structure is picked up by the robot during this process, the tile is placed at a neighboring node in a way that preserves connectivity.

Algorithm 1 Algorithm to form a straight line from any tile configuration by a single robot.

1: **procedure** MAKELINE
2: The robot moves S until there is no tile to step on.
3: **do**
4: Set flag *is_line* to TRUE.
5: *Tile searching phase:* at every step, until the robot can no longer move,
6: – if there is a neighboring tile at NW, SW, NE, or SE, set the flag *is_line* to FALSE;
7: – the robot repeatedly moves NW, SW or N (in this order of preference).
8: *Tile moving phase:* if *is_line* is FALSE, the robot picks up the tile at the current position, and moves it to the bottom of the adjacent column, starting at position SE.
9: **while** *is_line* is FALSE

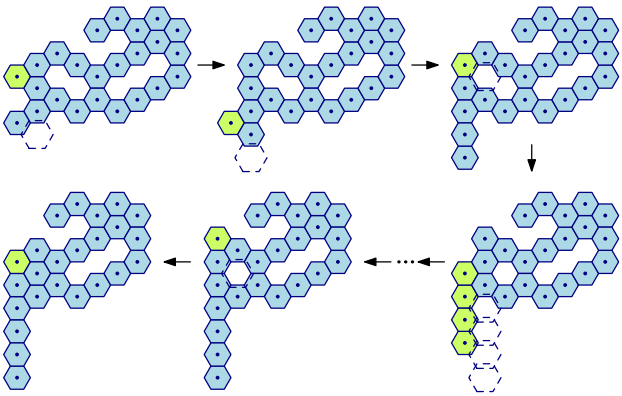


Figure 3: First several steps of the algorithm. The green tiles are moved to the positions marked by dashed frames.

3.1 Line Formation

We consider the problem of rearranging the tiles into a straight line. We will measure the efficiency of our algorithm in the number of *steps* (i.e., move actions) that the robot has to perform.

We present an algorithm for one robot to rearrange a tile configuration into a straight line in $O(n^2)$ steps. Throughout the algorithm we use the labels N, NE, SE, S, SW and NW (corresponding to cardinal directions) to refer to the six neighbors of the robot (see Figure 1). The pseudocode is given in Algorithm 1. At the beginning of every iteration of the algorithm, the robot is located at a locally most southern tile, i.e., there is no tile in the S direction. During one iteration of the algorithm, the robot finds a locally most north-western tile and moves it to the bottom of the column of tiles to the right from it. Figure 3 illustrates the first several iterations of the algorithm. To check whether the desired tile configuration has been achieved, the robot inspects neighboring tiles at each step in the search phase.

Theorem 2 *Following the procedure* MAKELINE*, a single robot can rearrange any tile configuration into a straight line in $O(n^2)$ steps.*

Proof. The correctness of the algorithm follows from the following observations: (i) the tile searching phase terminates in a locally most north-western tile, (ii) if there is more than one column in the tile configuration, the tile searching phase does not terminate in the topmost tile of the rightmost column, (iii) the tile moving phase does not disconnect the tile configuration and (iv) the algorithm terminates when a line is formed.

The first observation is obvious by the definition of the first phase of the algorithm. The second observation follows from the fact that the preference is given to the NW and SW directions when searching. If the target tile configuration has not yet been achieved, and the robot stops at some locally north-western tile, there must be tiles to the right from that position.

For the third observation, suppose that the tile moving phase disconnects the tile configuration. Let t be the locally most north-western tile being moved. The tile configuration can get disconnected after removing t only if there are neighboring tiles to NE and S of t, but no SE neighboring tile, since otherwise the neighboring tiles will still be locally connected after removing t. But in that case the tile t will be placed in the empty position at the SE neighbor and reconnect the neighboring NE and S tiles. Therefore, during the second phase of the algorithm the tile configuration does not get disconnected.

To prove the last claim, assign 2-dimensional coordinates to the centers of tiles. Let the x coordinate grow from left to right, and the y coordinate grow from top to bottom. Let 0 be the x-coordinate of a rightmost tile, thus the x-coordinate of any tile is not greater than 0. Consider the sum of the x-coordinates of all tiles $S = \sum_1^n x_t$. Initially, the value of S is negative, and it always increases by 1 after a tile is moved. The tile configuration is a straight line at $x = 0$, i.e., $S = 0$. No tiles will be moved to a position with an x-coordinate larger than 0. Therefore, the algorithm will terminate, and the terminal tile configuration will be a vertical straight line.

Finally, we show that the algorithm takes $O(n^2)$ steps. The preparation steps of the algorithm (line 2 of the Algorithm) take $O(n)$ steps. Consider the tile moving phase (line 8). Let the initial coordinates of some tile t be $(x_{t,0}, y_{t,0})$, and its final coordinates be $(0, y_{t,1})$. Each time the tile was moved, its coordinates were changed from some (x_t, y_t) to $(x_t + 1, y_t + \frac{1}{2} + c_t)$, where c_t is the number of tiles in the column, at the bottom of which the tile t was placed. The total number of steps the robot performed to move the tile from (x_t, y_t) to $(x_t + 1, y_t + \frac{1}{2} + c_t)$ is $1 + c_t$. Therefore, the total number of steps the robot performed to move the tile from its initial position to its final placement,

is $0-x_{t,0}+y_{t,1}-y_{t,0}-\frac{1}{2}(0-x_{t,0}) = -\frac{x_{t,0}}{2}+(y_{t,1}-y_{t,0})$. And the total number of steps the robot performed to move all the tiles is $\mathrm{s_{move}} = \sum_t \left(-\frac{x_{t,0}}{2}+(y_{t,1}-y_{t,0})\right) \leq \sum_t \frac{3}{2}n = O(n^2)$. Now, consider the tile searching phase (lines 5–7). Whereas the sum of the coordinates of the robot was increasing at every step in the tile moving phase, in the tile searching phase, the sum of the coordinates of the robot is decreasing at every step. More specifically, at each step of the tile moving phase, the sum of the coordinates of the robot increases by at most $\frac{3}{2}$, and at every step of the tile searching phase, the sum of the coordinates decreases by at least $\frac{1}{2}$. Thus, the total number of steps in the tile searching phase can be bounded in the following way: $\mathrm{s_{search}} < 3 \times \mathrm{s_{move}} + (x_0 + y_0) - \min_i(x_i + y_i)$, where (x_0, y_0) is the initial coordinates of the robot, and the value $\min(x_i+y_i)$ is taken over all possible placements of all tiles. As the initial tile configuration is connected, $(x_0 + y_0) - \min_i(x_i + y_i) = O(n)$, and $\mathrm{s_{search}} = O(n^2)$. Therefore, the total number of steps is $O(n^2)$. □

Note that it is not hard to see that $\Omega(n^2)$ steps are necessary to rearrange an arbitrary initial tile configuration into a straight line. If starting from a initial configuration with diameter $O(\sqrt{n})$, a constant fraction of the tiles have to be moved by a distance linear in n and thus, in total, $\Omega(n^2)$ move steps are necessary.

3.2 Triangle Formation

Once the robot has built a line, it can construct a triangle as follows: the robot picks up tiles from one end of the line and assembles them into a triangle at the other end following a zig-zag pattern, see Figure 4. It fills each *layer* of the triangle with tiles until it

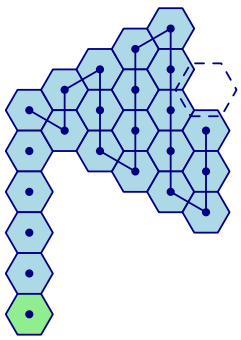


Figure 4: Triangle formation starting from a line

recognizes that the current position does not have a tile in the NW direction (when moving up), or that the current position does not have a tile in the SW direction (when moving down). In both cases, it starts a new layer of the triangle arrangement.

Theorem 3 *A single robot can rearrange any tile configuration into a triangle in $O(n^2)$ steps.*

It is not hard to see, using similar arguments as in the previous section, that this is asymptotically optimal.

4 Future Work

There are many directions for further research on shape formation with robots on tiles. First, we would be very interested to see how multiple robots can cooperate to speed up shape formation. Another obvious direction would be the formation of more complex shapes. Finally, it might be interesting to study an extension of the model in which each tile can have a state that can be read and modified by the robot.

Acknowledgments. This work was partially supported by DFG grant SCHE 1592/3-1. Irina Kostitsyna is supported by F.R.S.-FNRS and Fabian Kuhn is supported by ERC Grant No. 336495 (ACDC).

References

[1] A. Bonato and R. J. Nowakowski. *The Game of Cops and Robbers on Graphs*. AMS, 2011.

[2] S. Das. Mobile agents in distributed computing: Network exploration. *Bulletin of the European Association for Theoretical Computer Science*, 109:54–69, 2013.

[3] E. Demaine, M. Demaine, M. Hoffmann, and J. O'Rourke. Pushing blocks is hard. *Computational Geometry*, 26(1):21–36, 2003.

[4] Z. Derakhshandeh, R. Gmyr, A. W. Richa, C. Scheideler, and T. Strothmann. Universal shape formation for programmable matter. In *28th Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 289–299, 2016.

[5] F. V. Fomin and D. M. Thilikos. An annotated bibliography on guaranteed graph searching. *Theoretical Computer Science*, 399(3):236–245, 2008.

[6] F. Hurtado, E. Molina, S. Ramaswami, and V. Sacristán. Distributed reconfiguraiton of 2D lattice-based modular robotic systems. *Autonomous Robots*, 38(4):383–413, 2015.

[7] O. Michail and P. G. Spirakis. Terminating population protocols via some minimal global knowledge assumptions. *Journal of Parallel and Distributed Computing*, 81-82:1–10, 2015.

[8] M. J. Patitz. An introduction to tile-based self-assembly and a survey of recent results. *Natural Computing*, 13(2):195–224, 2014.

[9] A. Pelc. Deterministic rendezvous in networks: A comprehensive survey. *Networks*, 59(3):331–347, 2012.

[10] Y. Terada and S. Murata. Automatic modular assembly system and its distributed control. *International Journal of Robotics Research*, 27(3–4):445–462, 2008.

[11] D. Woods, H. Chen, S. Goodfriend, N. Dabby, E. Winfree, and P. Yin. Active self-assembly of algorithmic shapes and patterns in polylogarithmic time. In *Innovations in Theoretical Computer Science (ITCS)*, 2013.

Self-approaching paths in simple polygons

Prosenjit Bose* Irina Kostitsyna† Stefan Langerman†

1 Introduction

The problem of finding an optimal obstacle-avoiding path in a polygonal domain is one of the fundamental problems of computational geometry. Often a desired path has to conform to certain constraints. For example, a path may be required to be monotone [3], curvature-constrained [9], have no more than k links [16], etc. A natural requirement to consider is that a point moving along a desired path must always be getting closer to its destination. Such *radially monotone paths* appear, for example, in greedy geographic routing in network setting [10] and beacon routing in geometric setting [5]. A strengthening of a radially monotone path is a *self-approaching path* [12, 13, 1]: a point moving along a self-approaching path is always getting closer not only to its destination but also to all the points on the path ahead of it. There are several reasons to prefer self-approaching paths over radially monotone paths. First, unlike for a radially monotone path, any subpath of a self-approaching path is self-approaching. Therefore, if the destination is not known in advance and the desired path is required to be radially monotone, one will have to resort to using self-approaching paths. Second, the length of a radially monotone path can be arbitrarily large in comparison with the Euclidean distance between the source and the destination points, whereas self-approaching paths have a bounded detour.

In this paper we study self-approaching paths that are contained in a simple polygon. We consider the following questions:

- Given two points s and t inside a simple polygon P, does there exist a self-approaching s-t path inside P?
- Find the shortest self-approaching s-t path.
- Given a polygon P, test if it self-approaching, *i.e.*, that there exists a self-approaching path between any two points in P.

Related work. Self-approaching curves were first introduced in the context of online searching for a kernel of a polygon [12], and further studied in [13]. An equivalent definition of a self-approaching path is that for every point on the path there has to be a 90° angle containing the rest of the path. Aichholzer et al. developed a generalization of self-approaching paths for an arbitrarily fixed angle α instead of 90°. A relevant type of paths are increasing chords paths [18], which are self-approaching in both directions. The nice properties of self-approaching and increasing chords paths and their potential to be applied in network routing were recognized by the graph drawing community. As a result, a number of papers appeared in the recent years on self-approaching and increasing chords graphs [2, 8, 17].

This paper is organized in the following way. We introduce a few definitions and concepts in Section 2. In Section 3, we characterize a shortest self-approaching path between two points in a simple polygon. In Section 4 we present an algorithm to construct the shortest self-approaching path between two points if it exists, or to report that it does not exist, by assuming a model of computation in which we can solve certain transcendental equations. Finally, in Section 5 we present a linear-time algorithm to decide if a polygon is self-approaching, that is, if there is a self-approaching path between any two point of the polygon. Refer to the full version of this paper for the omitted proofs.

2 Preliminaries

A *self-approaching path* π in a continuous domain is a piece-wise smooth[1] oriented curve such that for any three points a, b, and c that appear on the curve in this order: $|ac| \geq |bc|$, where $|ac|$ and $|bc|$ are Euclidean distances.

Icking et al. showed the following *normal property* of a self-approaching path, that we will be using extensively in this paper,

Lemma 1 (the normal property [13]) *An s-t path π is self-approaching if and only if any normal to π at any point $a \in \pi$ does not cross $\pi(a, t)$.*

A normal h to a directed curve π at some point $a \in \pi$ defines two half-planes. Let the *positive half-plane* h^+ be the open half-plane which is congruent with the direction of π at point a. We can rephrase the normal property in the following way.

*Carleton University, Ottawa, Canada, jit@scs.carleton.ca

†Université libre de Bruxelles (ULB), Brussels, Belgium, {irina.kostitsyna,stefan.langerman}@ulb.ac.be

[1] Some previous works do not require the curve to be smooth. However in this paper we will be mostly considering shortest self-approaching paths, and thus the requirement on smoothness is justified.

Lemma 2 (the half-plane property) *An s-t path π is self-approaching if and only if, for any normal h to π at any point $a \in \pi$, the subpath $\pi(a,t)$ lies completely in the positive half-plane h^+.*

A *bend* of a self-approaching path π is a point of discontinuity of the first derivative of π.

A *reachable region* $\mathcal{R}(s) \subseteq P$, for a given point s in a polygon P, is a set of all points $t \in P$ for which there exists a self-approaching s-t path $\pi \in P$.

A *reverse-reachable region* $\mathcal{R}^{-1}(t) \subseteq P$, for a given point t in a polygon P, is a set of all points $s \in P$ for which there exists a self-approaching s-t path $\pi \in P$.

2.1 Involutes

In the following sections we will show that shortest self-approaching paths consist of straight-line segments, circular arcs, and involutes of circular arcs of some order. In the full version of this paper we introduce involute curves, and show the derivation of the following formula for an involute of a circle of order k:

$$I_k(\theta) = \sum_{0}^{\lfloor \frac{k}{2} \rfloor} (-1)^i a_{2i}(\theta) \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix} - \sum_{1}^{\lceil \frac{k}{2} \rceil} (-1)^{i-1} a_{2i-1}(\theta) \begin{pmatrix} -\sin\theta \\ \cos\theta \end{pmatrix},$$

where $a_i(\theta) = r_0 \frac{\theta^i}{i!} + c_1 \frac{\theta^{i-1}}{(i-1)!} + \cdots + c_i$.

Given a point $p_i(r_i, \varphi_i)$ for each involute I_i of order i (for all $1 \le i \le k$), the constants c_i can be found from the following equations:

$$\begin{aligned} r_i \cos(\theta_i - \varphi_i) &= a_0(\theta_i) - a_2(\theta_i) + \dots, \\ r_i \sin(\theta_i - \varphi_i) &= a_1(\theta_i) - a_3(\theta_i) + \dots, \end{aligned} \tag{1}$$

where θ_i is the parameter at which involute I_i passes through p_i. The length of the tangent segment from the point p_k to the involute I_{k-1} is $|a_k(\theta_k)|$.

3 Properties of a shortest self-approaching path

Next, we prove that a shortest self-approaching path is unique, and that the shortest self-approaching path consists of straight segments, circular arcs and involutes to the later pieces of the path.

We begin with proving several lemmas:

Lemma 3 *For any two points p_1 and p_2 (in this order) on a self-approaching s-t path π in $\mathbb{R}^2$, the perpendicular bisector of straight-line segment $\overline{p_1 p_2}$ does not intersect sub-path $\pi(p_2, t)$.*

Lemma 4 *Bends of a shortest self-approaching path in a simple polygon P form a subset of vertices of P.*

Lemma 5 *A shortest self-approaching s-t path in a simple polygon P cannot have an inflection point (or an inflection segment) that is interior to P.*

Define the *inflection* points of a directed geodesic path γ from s to t as the first points of the inflection segments of γ, *i.e.*, the set of last points in the maximal subchains of γ with the same direction of turn.

Lemma 6 *A shortest self-approaching path from s to t in a simple polygon P contains all the inflection points of the geodesic path from s to t.*

Consider two self-approaching paths π_1 and π_2 from s to t in a simple polygon P that do not have other points in common. Let γ be a geodesic path from s to t inside the area bounded by π_1 and π_2.

Lemma 7 *Geodesic path γ between two self-approaching paths π_1 and π_2 is also self-approaching.*

As a corollary to this lemma, for two self-approaching paths from s to t, a path, composed of geodesics in the areas bounded by subpaths of the two paths between each pair of consecutive intersection points, is also self-approaching. In other words, let $s = p_0, p_1, \dots, p_k, p_{k+1} = t$ be all the intersection points of π_1 and π_2 in the order they appear on π_1 and π_2. Observe, that the intersection points must appear in the same order along the both paths, otherwise there would exist three points on one of these paths for which the inequality in the definition of a self-approaching path would not be satisfied. Let γ_i be the geodesic from p_i to p_{i+1} in the area between two subpaths $\pi_1(p_i, p_{i+1})$ and $\pi_2(p_i, p_{i+1})$. Then,

Lemma 8 *The concatenation of the geodesics $\gamma = \gamma_0 \oplus \gamma_1 \oplus \cdots \oplus \gamma_k$ is self-approaching.*

Lemma 9 *For a given polygon P and points s and t in it, the convex hull of the shortest self-approaching path π^* is contained in the convex hull of any self-approaching s-t path π. That is $CH(\pi^*) \subseteq CH(\pi)$.*

The next theorem is a direct corollary of Lemma 8.

Theorem 10 *A shortest self-approaching s-t path is unique.*

Fig. 1 shows an example of a shortest self-approaching path inside a polygon. In the next two theorems and lemma we give its characterization.

Theorem 11 *The shortest self-approaching s-t path in a simple polygon consists of straight segments, circular arcs and circle involutes of some order.*

A path π is called *geodesically convex* when the shortest path connecting any two points of π lies completely on one (and the same side) of π.

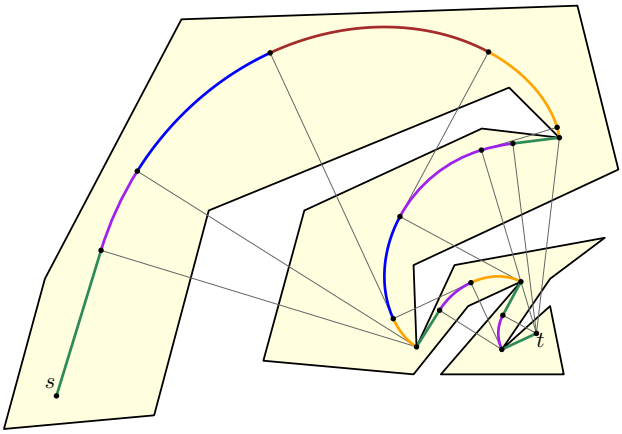


Figure 1: The shortest self-approaching path from s to t consists of line segments (green), circular arcs (purple), and involutes of a circle of some order (1st order in orange, 2nd—in blue, and 3rd—in brown).

Lemma 12 *The shortest self-approaching s-t path in a simple polygon P consists of geodesically convex paths between inflection points of the s-t geodesic.*

Theorem 13 *The shortest self-approaching s-t path in a simple polygon P consists of $O(n^2)$ segments. A shortest self-approaching s-t path may require $\Omega(n^2)$ segments.*

4 Existence of a self-approaching path

In this section we consider the question of testing whether, for given points s and t in a polygon P, they can be connected with a self-approaching path. In Theorem 11 we proved that a shortest self approaching path can consist of involutes of a circle of a high order, and in Section 2 we showed that such an involute is defined by a system of transcendental equations. In [15] Laczkovich proved a strengthening of Richardson's theorem, that states that the statement $\exists x : f(x) = 0$ is undecidable, where $f(x)$ is an expression generated by the rational numbers, the variable x, the operations of addition, multiplication, and composition, and the sine function. The Equations (1) that we need to solve to obtain formulas for the involutes are a special case of the class of expressions in Laczkovich's theorem. Nevertheless, it strongly suggests that an involute of a circle of order higher than one cannot be computed.

Next, we show an algorithm to test whether there exists a self-approaching path connecting two points s and t, and if so, to compute the shortest path, under the assumption that we can solve Equations (1). Subsequently, it may be possible to release this assumption, and modify the algorithm to build an approximation to the shortest path.

4.1 Shortest path algorithm

The proof of Theorem 11 is constructive. Let us assume that we can solve equations of the form as Equations (1) for an involute of order k in time $O(f(k))$, and evaluate the formula of the involute of order k for a given parameter θ in time $O(g(k))$. Then, we can decide if two points s and t can be connected by a self-approaching path, and we can construct the shortest path between the points. The outline of the algorithm:

Starting at t, move backwards along a geodesic s-t path γ. Maintain the convex hull CH of the final part of the shortest self-approaching path π^* to the destination t built so far. At every bend point p_ℓ:

(a) Calculate the appropriate branch of an involute I_{CH} of CH. If I_{CH} intersects the opposite boundary of the polygon, thus, cutting off s from t, report that a self-approaching path from s to t does not exist and terminate the algorithm.

(b) Otherwise, find a geodesic path γ_ℓ from the preceding inflection point of γ to p_ℓ in $P \backslash I_{CH}$, and add its last segment qp_ℓ as a prefix to π^*.

(c) Update CH. Repeat for the new bend point q, until s is reached. Report the found path π^*.

To obtain an algorithm with an optimal running time, there are a few considerations to take into account when constructing the shortest path. First, instead of unnecessarily calculating the whole involute I_{CH} until the intersection point with the boundary of P, and then discarding the part of it under the tangent line from q, its segments can be calculated one by one as needed until the tangent point. Second, to optimally test if I_{CH} intersects the opposite boundary of the polygon, we can maintain a shortest path tree that will allow us to build funnels from the opposite sides of the polygon boundary. Third, it is not necessary to construct the whole geodesic γ_ℓ to be able to compute its last segment qp_ℓ. Instead, we can move backwards along γ, vertex by vertex, until we reach a point from which a tangent to I_{CH} can be computed (possibly with adding new points along it).

Theorem 14 *The shortest self-approaching s-t path, if it exists, can be constructed in $O(k + \frac{n \log k}{\sqrt{k}}(g(\sqrt{k}) + f(\sqrt{k})))$ time, where k is the size of the output.*

5 Self-approaching polygon

A polygon is self-approaching, if for any two points there exists a self-approaching path connecting them.

Theorem 15 *Polygon P is self-approaching if and only if for any disk D centered at any point $p \in P$, the intersection $D \cap P$ has one connected component.*

Corollary 16 *Any self-approaching polygon is also increasing-chord.*

Next, we present an algorithm to test whether a given simple polygon P is self-approaching. From the proof of Theorem 15 it follows that the polygon P is self-approaching iff, for all edges e on the boundary of P directed in the counter-clockwise order, an area bounded between the two normals to e at its two end points in the right half-plane of e is free of ∂P. We call this area the *half-strip* of e. We will use this property to test efficiently if the polygon is self-approaching.

Let P be given as a set of points $p_0, p_1, \ldots, p_{n-1}$ in the counter-clockwise order around the boundary. We will start at p_0, move along the boundary in the counter-clockwise order and maintain the union of all the half-strips of the edges visited so far. More precisely, we will maintain the left and the right sides, ρ_l and ρ_r, of the hour-glass shape that is the union of the half-strips; ρ_l and ρ_r are convex polygonal chains. Store the segments of ρ_l and ρ_r as two lists, the last segments in the lists are infinite rays.

At every iteration of the algorithm, perform the following steps. Let p_i be the current point of the polygon P. The chain ρ_r contains the right side of the union of all the half-strips up to point p_{i-1}. Consider the next boundary segment $\overline{p_{i-1}p_i}$, and a perpendicular ray h_i at the point p_i. To update the chain ρ_r, do the following: For each segment $\overline{c_j c_{j+1}}$ on ρ_r,

- if $\overline{p_{i-1}p_i}$ intersects $\overline{c_j c_{j+1}}$, then report that P is not self-approaching and terminate;
- if h_i intersects $\overline{c_j c_{j+1}}$, calculate the intersection point c', and replace the first elements of the list ρ_r up to $\overline{c_j c_{j+1}}$ with two segments, $\overline{p_i c'}$ and $\overline{c' c_{j+1}}$; repeat for the next point p_{i+1}.

Traverse the boundary of polygon P twice in the counter-clockwise order, and then repeat the same algorithm traversing the boundary of P twice in the clockwise order. If none of the segments $\overline{p_{i-1}p_i}$ intersected a segment of ρ_r, report that P is self-approaching.

Theorem 17 *Given a simple polygon P with n vertices, the presented algorithm tests in $O(n)$ time if it is self-approaching.*

Acknowledgements. This work was begun at the CMO-BIRS Workshop on Searching and Routing in Discrete and Continuous Domains, October 11–16, 2015. I.K. was supported in part by the NWO under project no. 612.001.106, and by F.R.S.-FNRS.

References

[1] O. Aichholzer et al. Generalized self-approaching curves. *Discrete Applied Mathematics*, 109(1-2), 2001.

[2] S. Alamdari et al. Self-approaching Graphs. In *20th International Symposium on Graph Drawing (GD)*, 2012.

[3] E. M. Arkin, R. Connelly, and J. S. B. Mitchell. On monotone paths among obstacles with applications to planning assemblies. In *5th Annual Symposium on Computational Geometry (SCG)*, 1989.

[4] M. A. Bender and M. Farach-Colton. The LCA Problem Revisited. In *Latin American Symposium on Theoretical Informatics*, 2000.

[5] M. Biro et al. Beacon-Based Algorithms for Geometric Routing. In *13th Algorithms and Data Structures Symposium (WADS)*. 2013.

[6] B. Chazelle and D. P. Dobkin. Intersection of convex objects in two and three dimensions. *Journal of the ACM*, 34(1), 1987.

[7] B. Chazelle et al. Ray shooting in polygons using geodesic triangulations. *Algorithmica*, 12(1), 1994.

[8] H. Dehkordi, F. Frati, and J. Gudmundsson. Increasing-Chord Graphs On Point Sets. In *22nd International Symposium on Graph Drawing*. 2014.

[9] L. E. Dubins. On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents. *American Journal of Mathematics*, 79(3), 1957.

[10] J. Gao and L. Guibas. Geometric algorithms for sensor networks. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 370(1958), 2012.

[11] L. Guibas et al. Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons. *Algorithmica*, 2(1-4), 1987.

[12] C. Icking and R. Klein. Searching for the kernel of a polygon—a competitive strategy. In *11th Annual Symposium on Computational Geometry (SCG)*, 1995.

[13] C. Icking, R. Klein, and E. Langetepe. Self-approaching curves. *Mathematical Proc. of the Cambridge Philosophical Society*, 125(3), 1999.

[14] D. Kirkpatrick and J. Snoeyink. Computing common tangents without a separating line. In *4th International Workshop on Algorithms and Data Structures (WADS)*, 1995.

[15] M. Laczkovich. The removal of π from some undecidable problems involving elementary functions. *Proc. of the American Mathematical Society*, 131(07), 2003.

[16] J. S. B. Mitchell, C. Piatko, and E. M. Arkin. Computing a shortest k-link path in a polygon. In *33rd Annual Symposium on Foundations of Computer Science*. IEEE, 1992.

[17] M. Nöllenburg, R. Prutkin, and I. Rutter. On self-approaching and increasing-chord drawings of 3-connected planar graphs. *Journal of Computational Geometry*, 7(1), 2016.

[18] G. Rote. Curves with increasing chords. *Mathematical Proc. of the Cambridge Philosophical Society*, 115(01), 1994.

Routing in Simple Polygons

Matias Korman* Wolfgang Mulzer† André van Renssen‡,§ Marcel Roeloffzen‡,§ Paul Seiferth†
Yannik Stein† Birgit Vogtenhuber¶ Max Willert†

Abstract

A routing scheme $\mathcal{R}$ in a network $G = (V, E)$ is an algorithm that allows to send messages from one node to another in the network. We are first allowed a preprocessing phase in which we assign a unique *label* to each node $p \in V$ and a *routing table* with additional information. After this preprocessing, the routing algorithm itself must be local (i.e., we can only use the information from the label of the target and the routing table of the node that we are currently at).

We present a routing scheme for routing in simple polygons: for any $\varepsilon > 0$ the routing scheme provides a stretch of $1+\varepsilon$, labels have $\mathcal{O}(\log n)$ bits, the corresponding routing tables are of size $\mathcal{O}(\varepsilon^{-1} \log n)$, and the preprocessing time is $\mathcal{O}(n^2 + \varepsilon^{-1} n)$. This improves the best known strategies for general graphs by Roditty and Tov (Distributed Computing 2016).

1 Introduction

Routing is the act of sending a message from a node to its desired target. We would like to design a routing protocol that, for any network, can send messages from any node to any target. Although the problem is easy in small networks, it provides a significant challenge for large networks [1].

There are two key features of a routing scheme. First of all, it must be *local*: while the message is at a particular node, it can only use information stored in the memory of that node. In order to save storage, we would like that the amount of information that each node stores is relatively small. Second, the routing scheme should be *efficient*, meaning that the message should not travel much further than necessary.

A straight-forward solution is to explicitly store the whole network in each node. Routing can then be solved with a single source shortest path query, sending the message one step forward, and repeating the process until we eventually reach the final target. This is clearly local and efficient, but it requires a large amount of storage. Thus, the aim is to obtain some trade-off between the amount of information stored at each node, and the ratio between the distance travelled by messages and the shortest possible path for them in the network.

For general graphs, this problem has been well-studied since the 1980's [5, 6]. The most recent result is from Roditty and Tov [7] who developed a routing scheme for general graphs G with n nodes and m edges. The scheme has poly-logarithmic header size and routes a message from p to q on a path with length $\mathcal{O}(k\Delta + m^{1/k})$ for any integer $k > 2$, where Δ is the distance of the shortest path between p and q in G. Their routing tables use $mn^{\mathcal{O}(1/\sqrt{\log n})}$ total space, which is asymptotically optimal [5].

In this paper we provide a better algorithm, albeit for a specialized graph class: the visibility graphs of polygons. Given a simple polygon P of n vertices, we connect two vertices by an edge if and only if they can see each other (i.e., the segment connecting them is contained in P). Although many shortest path problems in polygons have been considered [2, 4], there are no routing schemes for visibility graphs of polygons. In this paper we present the first routing scheme for this graph class. For n vertices and any $\varepsilon > 0$, the routing scheme needs at most $\mathcal{O}(\varepsilon^{-1} \log n)$ bits for the routing table of each vertex and produces a routing path with stretch $1 + \varepsilon$. Therefore, for this class, our results provide a more efficient routing scheme than the one of Roditty and Tov [7].

2 Preliminaries

We model a network with an *undirected, connected* and *simple* graph $G = (V, E)$. We assume it is embedded in the Euclidean plane: a *node* $p = (p_x, p_y) \in V$ corresponds to a point and an edge $\{p, q\} \in E$ is represented by the segment $\overline{pq}$. We denote by $|\overline{pq}|$ the Euclidean distance between the points p and q and call $|\overline{pq}|$ the *weight* of the corresponding edge. The length of a shortest path in G connecting two points $p, q \in V$ is denoted by $d(p, q)$. From now on, we assume that G is the visibility graph of the n vertices of P (and thus, d encodes the *geodesic* distance in P).

There are several definitions for routing schemes for a network [3, 7]. In the following we introduce a restricted standard model that is comparable to the

*Tohoku University, Sendai, Japan. Partially supported by the ELC project (MEXT KAKENHI No. 12H00855 and 15H02665).

†Department of Computer Science, Freie Universität Berlin, Germany

‡National Institute of Informatics (NII), Tokyo, Japan.

§JST, ERATO, Kawarabayashi Large Graph Project.

¶Institute for Software Technology, Graz University of Technology, Graz, Austria.

other definitions. Each node in G has a unique tag, called *label*, that identifies the node in the network, as well as some additional information stored in a *routing table*. Starting at any node $p \in V$, the routing scheme uses only the information of p's routing table (and the target's label) to compute a node adjacent to p where the message is forwarded to. This process is repeated until it reaches the target.

Definition 2.1 *A routing scheme $\mathcal{R} = (l, \rho, f)$ of a graph G consists of the following elements: a label $l(p) \in \{0,1\}^*$ and a routing table $\rho(p) \in \{0,1\}^*$ for each node $p \in V$, as well as a routing function $f\colon V \times \{0,1\}^* \to V$.*

The transition function f models the behaviour of the routing scheme. For any two nodes $p, q \in V$, consider the sequence of points given by $p_0 = p$ and $p_i = f(p_{i-1}, l(q))$ for $i \geq 1$ (i.e., the nodes visited in the routing scheme from p to q). We say that a routing scheme $\mathcal{R}$ is *correct* if for any $p, q \in V$ there exists a $k = k(p,q) \geq 0$ such that $p_k = q$ and $p_i \neq q$ for $i < k$. We call $p_0, p_1, \ldots, p_k$ the *routing path* between p and q. The *routing distance* between p and q is defined as $d_\rho(p,q) = \sum_{i=1}^{k} |\overline{p_{i-1}p_i}|$.

The quality of the routing scheme is measured by several parameters:

- *label size* $L(n) = \max_{|V|=n} \max_{p \in V} |l(p)|$,
- *table size* $T(n) = \max_{|V|=n} \max_{p \in V} |\rho(p)|$,
- *stretch* $\zeta(n) = \max_{|V|=n} \max_{p \neq q \in V} \frac{d_\rho(p,q)}{d(p,q)}$,
- and *preprocessing time* (i.e., time spent in computing the labels and routing tables).

Let P be a polygon with n vertices (which need not be in general position). Two points $p, q \in P$ can *see each other* if and only if $\overline{pq} \subset P$. Note that p and q can see each other even if the line segment $\overline{pq}$ touches the boundary of P. The *visibility graph* $\mathrm{VG}(P)$ of P is the graph whose vertex set is the vertex set of P. Two vertices are connected with an edge in $\mathrm{VG}(P)$ if in P they see each other. In this paper we show that for any $\varepsilon > 0$ we can construct a routing scheme with $\zeta(n) = 1 + \varepsilon$, $L(n) = \mathcal{O}(\log n)$ and $T(n) = \mathcal{O}(\varepsilon^{-1} \log n)$. The preprocessing time is $\mathcal{O}(n^2 + \varepsilon^{-1} n)$.

3 Cones in Polygons

Let P be a polygon with n vertices and let $t > 2$. Our approach is inspired by the Yao Graph construction [8].

Let p be a vertex of the polygon, p' the clockwise next vertex of P, α be the inner angle at p, and $\varepsilon_0 := \frac{2\pi}{\alpha t}$. We denote with r the *ray* emanating from p through p'. Next, we rotate this ray as follows: let

$$r_i(p) := \text{rotate } r \text{ clockwise by angle } \alpha \cdot \min(i \cdot \varepsilon_0, 1)$$

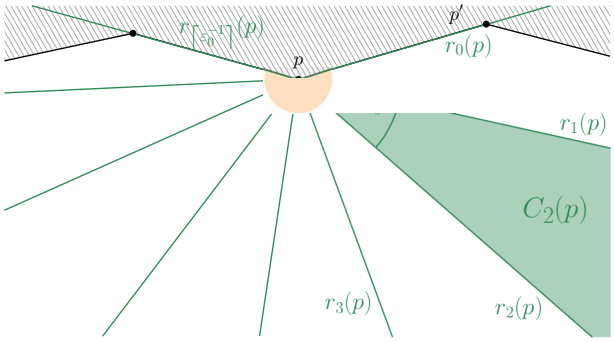


Figure 1: The cones and rays of a vertex p with inner angle α.

for $i \in \{0, 1, \ldots, \lceil \varepsilon_0^{-1} \rceil\}$. Let $C_i(p)$ be the closed *cone* with apex p and boundary $r_{i-1}(p)$ and $r_i(p)$ (see Figure 1). We also set $\mathcal{C}(p)$ as the set containing all such cones (that is, $\mathcal{C}(p) = \{C_i(p) \mid 1 \leq i \leq \lceil \varepsilon_0^{-1} \rceil\}$). By construction, the apex angle of each cone is at most $\alpha\varepsilon_0 = 2\pi/t$ and hence all cones are convex.

Lemma 3.1 *Let p be a vertex in P and $\{p, q\}$ an edge of $\mathrm{VG}(P)$ in the cone $C_i(p)$. Furthermore, let s be the closest vertex in $C_i(p)$ to p. Then the following inequality holds:*

$$d(s,q) \leq |\overline{pq}| - \left(1 - 2\sin\frac{\pi}{t}\right)|\overline{ps}|.$$

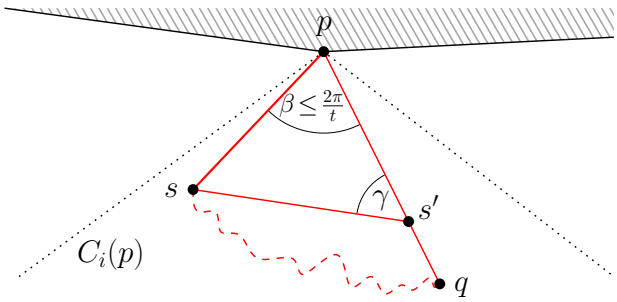


Figure 2: Illustration of Lemma 3.1. The points s and s' have the same distance to p. The dashed line represents the shortest path from s to q.

Proof. Let s' be the point on the line segment $\overline{pq}$ such that $|\overline{ps'}| = |\overline{ps}|$ (see Figure 2). Since p can see q, also s' can see q. Since s is the closest vertex to p in the cone, s can see s'. Now the triangle inequality yields $d(s,q) \leq |\overline{ss'}| + |\overline{s'q}|$. Let β and γ be the angles at p and s' in the triangle $\Delta(p, s, s')$. Since $\Delta(p, s, s')$ is in $C_i(p)$, we have $\beta \leq 2\pi/t$. Further, $\Delta(p, s, s')$ is isosceles and thus $\gamma \geq \pi/2 - \pi/t$. Applying the sine law and $\sin 2x = 2 \sin x \cos x$, we get

$$|\overline{ss'}| \leq 2|\overline{ps}| \sin\frac{\pi}{t}.$$

Together with $|\overline{s'q}| = |\overline{pq}| - |\overline{ps'}| = |\overline{pq}| - |\overline{ps}|$, the triangle inequality from before gives the desired estimation. □

4 The Routing Scheme

Let $\varepsilon > 0$ and P be a polygon with vertices $p_1, \ldots, p_n$ in this order. We describe a routing scheme for $\mathrm{VG}(P)$ with stretch $1+\varepsilon$. For each each vertex p, we will partition the other vertices into intervals and assign one interval to each cone. Given a target vertex q, we look for the cone $C_i(p)$ whose associated interval contains q, and transmit the message to the nearest neighbour in $C_i(p)$.

Preprocessing In the preprocessing phase we do as follows. First, we assign to each vertex p_j of P the binary representation of j as its label, Which needs $\mathcal{O}(\log n)$ bits. For the routing table of a vertex p, we first compute the visibility polygon $\mathrm{vis}(p)$. This computation provides a sequence of consecutive points $v_0 v_1 \ldots v_k$ with $p = v_0 = v_{k+1}$. Each point of this sequence is either a vertex of P or the first proper intersection of a ray from p towards a reflex vertex. Notice that, as we walk clockwise along the visibility polygon, the angle spanned by the ray $r_0(p)$ and the edge $\{p, v_j\}$ increases monotonically. We set $t := \pi / \arcsin\left(0.5/\left(1+\varepsilon^{-1}\right)\right)$ and use the subdivision of $\mathrm{vis}(p)$ described in Section 3. This subdivision provides a set $\mathcal{C}(p)$ with a certain number of cones. The following obvious lemma specifies this number.

Lemma 4.1 *We have* $t \leq 2\pi\left(1+\varepsilon^{-1}\right)$.

The ray $r_i(p)$ intersects $\mathrm{vis}(p)$ at one or more points. Let z_i be the intersection point that is closest to p and e_i the edge of P containing it. All points z_i and edges e_i can be obtained by going once through the sequence $v_0 v_1 \ldots v_k$ (see Figure 3).

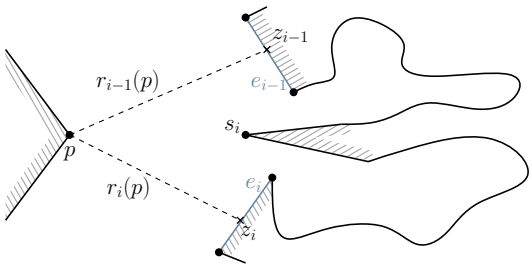


Figure 3: The boundaries of $C_i(p)$ hit the boundary of P in the points z_{i-1} and z_i. The vertex s_i is the point in $C_i(p)$ with shortest distance to p.

For all cones $C_i(p) \in \mathcal{C}(p)$, we start from z_{i-1}, walk clockwise along the boundary of P until we meet z_i, and collect all the visited vertices. If we take the indices modulo n, the collection forms an interval called $I(i)$. Among $I(i)$ we look for the vertex s_i with smallest distance to p. We store the interval boundaries of $I(i)$ together with the label of the vertex s_i in the routing table of p. This requires $\mathcal{O}(\log n)$ bits as well. Hence, by Lemma 4.1, the size of each routing table is $\mathcal{O}(\varepsilon^{-1} \log n)$ bits.

Routing phase The routing strategy is simple: when routing from a vertex p to a target q, we search in the routing table of p for the index i whose associated interval $I(i)$ contains the label of q, and then transmit the message to s_i. The following lemma proves that the algorithm is well defined.

Lemma 4.2 *Let p, q be two vertices of P and (p, q') the first edge on the shortest path from p to q. If $q \in I(i)$, then $q' \in C_i(p)$.*

Proof. Suppose that $q' \notin C_i(p)$. Since q is in $I(i)$, the shortest path π from p to q has to cross $\overline{pz_{i-1}}$ or $\overline{pz_i}$ at least twice. The first intersection is p itself. Let z be the last intersection and let π' be the subpath of π from p to z via q'. By the triangle inequality, $|\overline{pz}|$ is strictly smaller than the length of π' (see Figure 4). Thus, we can find a shortcut from p to z and hence a shorter path from p to q. □

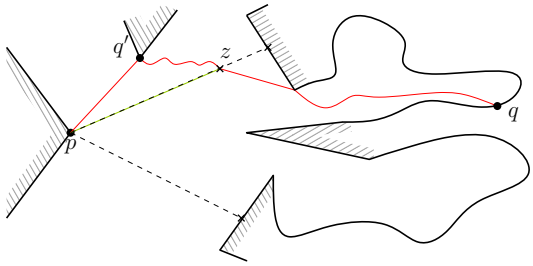


Figure 4: The red line is the "shortest" path from p to q with q' as first step, whereas the green dashed line represents a shortcut from p to z.

5 Analysis

The aim of this section is to show that for any polygon P and $\varepsilon > 0$ the stretch $1+\varepsilon$ is always preserved. First, we show that our routing strategy decreases the distance to the target.

Lemma 5.1 *Let p and q be two vertices in P, and let s be the next vertex computed by the routing scheme from p to q. Then we have $d(s,q) \leq d(p,q) - |\overline{ps}|/(1+\varepsilon)$.*

Proof. Our routing strategy routes from p to a vertex $s = s_i$ that is the closest in some cone $C_i(p)$. By Lemma 4.2, we know that the next vertex q' on the shortest path from p to q is also contained in $C_i(p)$. Thus, by Lemma 3.1 and the triangle inequality we obtain

$$\begin{aligned} d(s,q) &\leq d(s,q') + d(q',q) \\ &\leq |\overline{pq'}| - \left(1 - 2\sin\frac{\pi}{t}\right)|\overline{ps}| + d(q',q) \\ &= d(p,q) - \left(1 - 2\sin\frac{\pi}{t}\right)|\overline{ps}| \\ &= d(p,q) - |\overline{ps}|/(1+\varepsilon). \end{aligned}$$

□

Since the distance to the target decreases at each step, this implies that our routing scheme terminates. We now bound the stretch of the routing scheme.

Lemma 5.2 *Let p and q be two vertices of P. Then we have $d_\rho(p,q) \leq (1+\varepsilon)d(p,q)$, where $d_\rho(p,q)$ is the length of the routed path.*

Proof. Let $\pi = p_0 p_1 \dots p_k$ be the path from $p = p_0$ to $q = p_k$ computed by the routing scheme. By Lemma 5.1 we have $d(p_{i+1},q) \leq d(p_i,q) - |\overline{p_i p_{i+1}}|/(1+\varepsilon)$. Thus, we have

$$\begin{aligned} d_\rho(p,q) &= \sum_{i=0}^{k-1} |\overline{p_i p_{i+1}}| \\ &\leq (1+\varepsilon)\sum_{i=0}^{k-1} (d(p_i,q) - d(p_{i+1},q)) \\ &= (1+\varepsilon)\,(d(p_0,q) - d(p_k,q)) \\ &= (1+\varepsilon)d(p,q) \end{aligned}$$

since $p_0 = p$ and $p_k = q$. This finishes the proof for the bound on the stretch. □

Thereby, we obtain our main theorem.

Theorem 5.3 *Let P be a simple polygon with n vertices. For any $\varepsilon > 0$ we can preprocess P into a routing scheme for $\mathrm{VG}(P)$ with labels of $\mathcal{O}(\log n)$ bits and routing tables of $\mathcal{O}(\varepsilon^{-1}\log n)$ bits. For any two vertices $p, q \in P$, the scheme produces a routing path with stretch $\leq (1+\varepsilon)d(p,q)$. The preprocessing time is $\mathcal{O}(n^2 + \varepsilon^{-1}n)$.*

Proof. The stretch and size bounds follow from previous arguments. We focus on the preprocessing time bound. For any vertex $p \in P$ let L be the sequence of vertices $v_0 v_1 \dots v_k$ of $\mathrm{vis}(p)$ calculated in time $\mathcal{O}(n)$. Using L, we can find all intersection points z_i and corresponding edges e_i of P in time $\mathcal{O}(n + \varepsilon^{-1})$. Thus, for each ray $r_i(p)$, we can find the boundaries of e_i in amortized constant time. Once the edges e_i are computed, we can find the interval boundaries of $I(i)$ in constant time. The point within the interval $I(i)$ with smallest distance to p can be found by going once through $I(i)$ in $\mathcal{O}(|I(i)|)$ time. For all cones of one vertex, this step takes $\mathcal{O}(n+\varepsilon^{-1})$ in total. In the end, we do the same procedure for all vertices and obtain the running time $\mathcal{O}(n^2 + \varepsilon^{-1}n)$. □

6 Conclusion

We still have various open questions for the routing schemes for polygons. First of all, it would be interesting, if there is a routing scheme that approximates the hop-distance in polygons, where each pair of adjacent vertices has edge weight 1. Using our routing scheme we can find examples, where the stretch is in $\Omega(n)$. Further, it would be interesting to know whether the preprocessing time or the size of the routing table can be improved, perhaps using a recursive strategy.

The routing scheme extends to polygonal domains with h holes: we now need to apply the same strategy to each vertex p, cone $C_i(p)$ and the $h+1$ simple polygons that form the boundary of P. This increases the size of the routing table to $\mathcal{O}(\varepsilon^{-1}h\log n)$ and preprocessing time to $\mathcal{O}(n^2\log n + hn^2 + \varepsilon^{-1}hn)$. This is impractical for large values of h, and thus we wonder whether a better strategy exists for this case.

References

[1] Silvia Giordano and Ivan Stojmenovic. Position based routing algorithms for ad hoc networks: A taxonomy. In *Ad hoc wireless networking*, pages 103–136. Springer-Verlag, 2004.

[2] John Hershberger and Subhash Suri. An optimal algorithm for Euclidean shortest paths in the plane. *SIAM J. Comput.*, 28(6):2215–2256, 1999.

[3] Haim Kaplan, Wolfgang Mulzer, Liam Roditty, and Paul Seiferth. Routing in unit disk graphs. In *Proc. 12th Latin American Symp. Theoretical Inf. (LATIN)*, pages 536–548, 2016.

[4] Joseph S. B. Mitchell. A new algorithm for shortest paths among obstacles in the plane. *Annals of Mathematics and Artificial Intelligence*, 3(1):83–105, 1991.

[5] David Peleg and Eli Upfal. A trade-off between space and efficiency for routing tables. *J. ACM*, 36(3):510–530, 1989.

[6] Liam Roditty and Roei Tov. New routing techniques and their applications. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing*, pages 23–32. ACM, 2015.

[7] Liam Roditty and Roei Tov. Close to linear space routing schemes. *Distributed Computing*, 29(1):65–74, 2016.

[8] Andrew Chi-Chih Yao. On constructing minimum spanning trees in k-dimensional spaces and related problems. *SIAM J. Comput.*, 11(4):721–736, 1982.

Kinetic All-Pairs Shortest Path in a Simple Polygon*

Yago Diez† Matias Korman† André van Renssen‡ Marcel Roeloffzen‡ Frank Staals§

Abstract

We provide a simple data structure to compute all-pairs shortest paths of a given collection of n sites in a simple polygon of m corners. The structure can be maintained as the sites move (in a linear fashion) within the domain. For each event in which the combinatorial structure of the shortest path changes, we spend $O(\log nm)$ time updating our structure. We give upper and lower bounds on the number of such changes, proving that our structure is efficient.

1 Introduction

Motion data is common in many modern applications. We track physical objects (animals, particles, robots) as well as virtual ones in games and simulations. Sometimes the purpose is collecting data for later analysis, but often it must be processed directly to maintain important properties of a larger system. Generally these entities do not operate in a vacuum and environmental concerns should be taken into account. Oftentimes the environment is inherently important to the properties being tracked as obstacles may hamper navigation [1] or limit visibility [8].

Given the number of applications, much research has been done towards maintaining properties of moving points. Much of this work, however, considers moving entities in an open space. Only a handful of results consider problems where obstacles form an inherent part of the problem. Most notably, maintaining the visibility region of a moving point [7, 2]. In this paper we focus on structures related to the shortest paths for points moving inside a simple polygon. We provide a data structure following the KDS-framework (see below) that maintains the shortest path between pairs of points. As an application, we obtain a KDS to maintain the visibility graph of a set of moving points.

KDS-framework. Our data structures are based on the Kinetic Data Structures (KDS) framework introduced by Basch et al. [3]. In this framework motions are assumed to be known in advance in order to accurately predict when changes in the structure occur and then adjust accordingly. Each KDS maintains a set of *certificates* that together certify that the KDS correctly represents the target structure. Typically these certificates involve a few objects each and represent some simple geometric primitive. For example a certificate may indicate that three sites p, q, r form a clockwise oriented triangle $\Delta(p, q, r)$. As the sites move, these certificates may become invalid, requiring the KDS to update by potentially repairing the target structure and creating new certificates. Such a certificate failure is called an event. An *external event* is any event that changes the target structure, whereas every event is considered an *internal event*. Performance of a KDS is measured using four measures. A KDS is considered *compact* if it requires little space, generally close to linear, *responsive* if each event is processes quickly, generally in polylogarithmic time, *local* if each site partipates in few events, and *efficient* if the ratio between external and internal events is small, generally polylogarithmic. (Note that for efficiency it is common to compare the worst-case number of events for either case.) This framework has been used to maintain many structures; see the survey by Guibas et al. [4].

Our Results. In Section 2 we present a simple KDS for maintaining the combinatorial structure of the shortest path $\pi(p_i, q_i)$ between pairs of points p_i and q_i moving linearly inside a simple polygon P. Aside from a linear size representation of P, our data structure uses only $O(1)$ space per pair of points. It processes $O(m)$ events per pair, each taking $O(\log km)$ time, where m is the number of vertices of P and k is the number of pairs tracked. This thus gives us a KDS to maintain all-pairs shortest paths of a set S of n moving points using $O(m + n^2)$ space, and processing $O(n^2 m)$ events in $O(\log nm)$ time each. By tracking when the shortest path, the *geodesic*, is a single line segment we can then also maintain which pairs of points are mutually visible. That is, we can maintain the visibility graph $G = (S, \{(p, q) \mid q \text{ visible from } p\})$ of S. In Section 3 we show that for these scenarios our KDS is efficient, that is, we show that the number of visibility events, and thus the number of changes in the combinatorial structure of the geodesic, may be $\Omega(n^2 m)$.

Related Work. We are aware of only two other results involving distances, and points moving amidst

*M. K. was supported in part by the ELC project (MEXT KAKENHI No. 12H00855 and 15H02665).

†Tohoku University, Sendai, Japan. {yago, mati}@dais.is.tohoku.ac.jp

‡National Institute of Informatics (NII), Tokyo, Japan. {andre,marcel}@nii.ac.jp, JST, ERATO, Kawarabayashi Large Graph Project.

§MADALGO, Aarhus University. f.staals@cs.au.dk

obstacles: the work of Karavelas and Guibas [6], and the work of Aronov et al. [2]. Karavelas and Guibas describe how to maintain a constrained Delaunay triangulation of a set of moving points, and show how this can be used to maintain nearest neighbors. Aronov et al. provide an $O(m)$ space structure that maintains the *shortest path tree* of a point p moving inside a simple polygon P with m vertices. That is, their data structure maintains the shortest paths from p to all vertices of the polygon. Our data structure shares similarities with the approach of Aronov et al.. The main difference is that in our approach, both end-points of a shortest path are moving, whereas in Aronov et al. [2] one of the end-points, a vertex of P, is static.

Aronov et al. [2] use their data structure to maintain the visibility region of the moving point p. There are some additional results specifically for this problem. In particular, Rivière [7] presents two kinetic data structures for this problem, in one of which the vertices of the polygon are allowed to move as well. This data structure has size $O(m^2)$, and processes each event in $O(\log m)$ time. No analysis of the number of events is given.

2 Maintaining geodesics between moving points

We start by preprocessing the polygon P for two point shortest path queries [5]. This takes $O(m)$ time, and requires $O(m)$ space. We now show that we can maintain the geodesic $\pi(p, q)$ between two points p and q moving inside P with only $O(1)$ additional space. Throughout the movement of p and q we process $O(m)$ events.

Maintaining $\pi(p, q)$ explicitly may require $\Omega(m)$ total storage, so instead, we maintain the first and last edge of the path. We show that this is sufficient to implicitly maintain the path, in $O(1)$ time, and report the full path in time proportional to its length.

Denote the start- and end-point of the motion of p (q) by p_s (q_s) and p_e (q_e), respectively. We maintain the first edge on the geodesic from p to both q_s and q_e and from q to both p_s and p_e. We call these the funnel geodesics between p and q. Here a *funnel* is the union of all shortest paths from a single site to every point on a segment. In the following we first show how to maintain the funnels from the moving site p to q_sq_e and from q to p_sp_e and then provide details of how to maintain the path between the two moving sites.

2.1 The path between a static and a moving site

Each of the funnels consists of two geodesics from a moving point to a static point. Here we focus on maintaining the path from a site p moving from p_s to p_e and a static point, say q_s (the case for q_e is analogous). The simple, yet crucial observation, that allows us to maintain $\pi(p, q_s)$ efficiently, is that this path always consists of a single line segment followed by a tail-section of either $\pi(p_s, q_s)$ or $\pi(p_e, q_s)$.

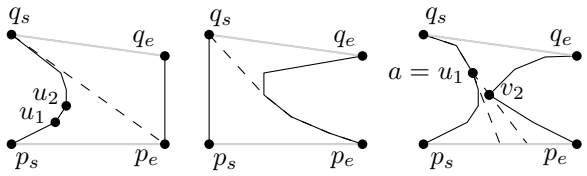


Figure 1: There are three cases to consider in the funnel from p to a fixed point q_s.

First consider the simplified case of Fig. 1 (left) where the path from p_s to q_s is a convex chain and p_e can directly see q_s. Let u_1, u_2 be the first two vertices that are traversed from p on the path to q_s. (If $u_1 = q_s$ the path no longer changes, so assume this is not the case.) Now, during the movement of p, this geodesic does not change while the triangle $\Delta(p, u_1, u_2)$ remains counter-clockwise oriented. Whenever the three points become aligned we have a *detachment event*: the topology of the shortest path between p and q_s changes, as the path does not pass through u_1 anymore. Thus, we remove it from the structure, update the values u_1 and u_2 (as the first and second vertices traversed in the path from the *current* position of p to q_s), and continue until the next detachment event. Eventually, all vertices of the path are removed and the path from p to q_s becomes segment $\overline{pq_s}$. This situation will continue until p reaches p_e (recall that we assumed that p_e sees q_s).

A similar situation occurs when the path from p_s to q_s is a single edge and the path from p_e to q_s is a convex chain as in Fig. 1 (middle). Let v_2 be the first vertex traversed on the path from $u_1 = q_s$ to p_e. We have an *attachment certificate* that states that triangle $\Delta(p, u_1, v_2)$ is clockwise oriented. If this certificate breaks, we add v_2 to the path from q_s to p, and a new attachment certificate can be found by computing the first edge on the shortest path from v_2 to p_e.

Now the more general case where neither p_s nor p_e have direct visibility of q_s is essentially a combination of the previous two (see Fig. 1 right). Consider the pseudotriangle between p_s, q_s, p_e with apex a where the two paths from p_s and p_e to q_s merge. Since the portion of the path from a to q_s is present in any path from p to q_s, it is equivalent to track the path from p to a. Further note that the geodesics from a to both p_s and p_e form two convex chains. Consider a time t where a is directly visible from p. For the movement before time t we can use the detachment certificates and after time t we can use the attachment certificates. So the main issue is detecting when we are directly connected to the apex a. We could explicitly compute a while preprocessing, but this could require $\Omega(n)$ preprocessing time. Instead we compute both

certificates and compute which is broken first. Whenever a detachment certificate is broken, we update u_1, u_2, and v_2 (since the definition of v_2 depends on u_1). In particular, this can change the point in time in which the attachment certificate would break. The first time we break the attachment certificate is when p has a direct edge to the apex a. From this instant on, we can consider only attachment certificates.

Lemma 1 *Using $O(m)$ preprocessing time and space we can maintain the shortest path from a site p moving linearly from p_s to p_e to a static point within P. Along the movement of p at most $O(m)$ events are processed, each one needs $O(\log m)$ time.*

Proof. As described above at each event we spend $O(1)$ time to update the shortest path. To compute new certificates, we use Guibas and Hershberger's two point query data structure for simple polygons [5], which allows us to query for the first $O(1)$ edges on a geodesic from any point inside the polygon to another in $O(\log m)$ time after $O(m)$ time preprocessing.

The bound on the total number of events follows from the fact that the shortest path only changes when p crosses an edge of the shortest path map of the static points within P, which is a $O(m)$ complexity straight-line subdivision of P. □

2.2 The path between two linearly moving sites

To maintain the geodesic between two moving sites p and q, we track four pairs of geodesics between moving and static points. Namely, we track the geodesics between p and q_s, q_e and between q and p_s, p_e. First we characterize when the geodesic $\pi(p, q)$ can change.

Lemma 2 *Let p and q be two sites moving linearly from p_s and q_s to p_e and q_e, respectively. The path $\pi(p, q)$ changes only when either p gains or loses visibility to q, if the funnel from p to $q_s q_e$ or the funnel from q to $p_s p_e$ changes.*

Proof. Let π_0 and π_1 be the geodesic before and after the change, respectively. By continuity of the shortest path, at most one vertex has been added between the two. Moreover, this vertex must be either the first or last in the geodesic from p to q. Without loss of generality we assume that a vertex r was added to the geodesic, and that p connects directly to r. That is, $\pi_0 = (p, u_0, \dots q)$ and $\pi_1 = (p, r, u_0, \dots q)$. If $u_0 = q$ then p and q were originally visible and thus we have a loss of visibility event as claimed. Otherwise, p, r and u_0 just broke an attachment certificate. □

This characterization allows us to maintain the geodesic between the two moving points.

Lemma 3 *Using $O(m)$ preprocessing time and space we can maintain the shortest path between two linearly moving sites within a polygon P of m vertices. Along the movement of both sites at most $O(m)$ events are processed, each one needs $O(\log m)$ time.*

Proof. Changes in the funnels from p and q are detected using the data structure detailed in Section 2.1. It remains to show how to detect visibility events. Since at any event only one vertex can be added to or removed from the path, we need not worry about visibility events until only one vertex remains on the path from p to q. In order for p to be directly visible from q, the funnel from p to $q_s q_e$ must have p as its apex, since otherwise it has no visibility on the whole edge $q_s q_e$. Furthermore when p is the apex, we can find the section of $q_s q_e$ that is directly visible to p by extending the first edge on the paths from p to q_s and q_e. Hence, this interval is defined by p and the first two points on the paths to q_s and q_e which do not change unless a funnel event happens. While these two funnel points do not change, we can compute certificates for the visibility events by computing when q enters or leaves this moving section of $q_s q_e$ that is visible to p. □

This allows us to track the geodesic between any pair of moving points using only $O(1)$ additional space and handling $O(m)$ events. For maintaining all shortest paths among a set of points we thus get:

Theorem 4 *Let P be a simple polygon with m vertices, and let S be a set of n sites each moving along a line segment in P. There is a KDS of size $O(m + n^2)$ that maintains the shortest path between any pair of sites $p, q \in S$ that processes at most $O(n^2 m)$ events, each in $O(\log m)$ time. The data structure can be built in $O(n^2 \log m + m)$ time.*

Note that during the full movement of p, our approach from Section 2.1 essentially runs the algorithm to report a geodesic on the funnel geodesics $\pi(p, q_s)$ and $\pi(p, q_e)$ (where events from different funnels may be interleaved). This means that, at any time, we can report these funnel geodesics by simply running the remainder of the computation. Since the geodesic between p and q consists of constantly many pieces of these funnel geodesics, it follows we can also report their geodesic in time proportional to its length.

We can now easily track when the geodesic between p and q is a single line segment, and thus p and q are visible, and thus we obtain:

Corollary 5 *Let P be a simple polygon with m vertices, and let S be a set of n sites each moving along a line segment in P. There is a KDS of size $O(m + n^2)$ that maintains the visibility graph of S, and processes $O(n^2 m)$ events, each in $O(\log nm)$ time. The data structure can be built in $O(n^2 \log m + m)$ time.*

Our data structures are responsive, and, as we show in Section 3, efficient, but it is not local or compact. However, it seems hard, maybe even infeasible, to track all $\Theta(n^2)$ distances using only $O(n \log^c n)$ space.

3 A bound on the number of visibility events

Here, we present a lower bound for the number of combinatorial changes to the geodesics between all pairs of n sites in a simple polygon of m vertices. We prove that the number of visibility events, that is, the number of times a site can gain or lose visibility to another is lower bounded by $\Omega(n^2 m)$. Since each visibility event is also a change in the geodesic the lower bound for geodesic changes follows.

Lemma 6 *A set S of n linearly moving sites in a simple polygon of m vertices can have $\Omega(n^2 m)$ visibility events.*

Proof. Consider two sites p and q, initially vertically aligned with p above q. Now p moves down vertically and q horizontally, both at unit speed. We create a polygon so that p and q gain and lose visiblility $\Omega(m)$ times. Let $t_0, \ldots t_m$ be m arbitrary moments of time during this movement in increasing order, with t_0 the time where p, q are at their initial positions and t_m such that p is still vertically above q. Now virtually draw a solid line segment between points $p(t_i)$ and $q(t_i)$ for even i and a dashed line for odd i as in Fig. 2.

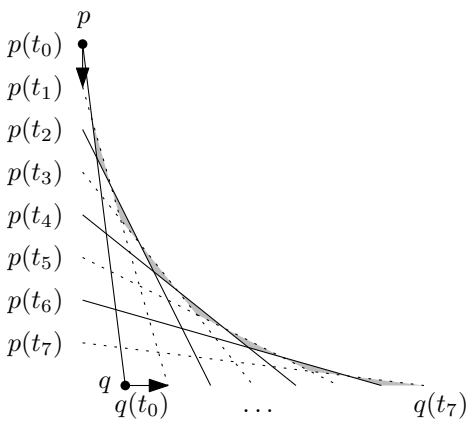


Figure 2: A lower bound on the number of events to maintain the visibility graph. Solid lines represent moments when p and q are visible and the dashed lines represent moments in time when they are not.

Observe that the upper envelope of the solid and dashed lines alternatingly has a dashed and solid segment. Now create a polygon vertex just below each dashed segment, but above the solid segments and connect these in the order that the corresponding segments appear on the upper envelope. This polygonal chain blocks visibility from p to q at time t_i for odd i, but does not do so for each time t_i for even i, so p and q gain and lose visibility $\Omega(m)$ times. The chain can easily be extended to be a polygon containing the full motions of p and q.

To prove the lower bound of $\Omega(n^2 m)$, we replace the single sites p and q with two sets of $n/2$ sites each. When these sets of sites follow trajectories that are sufficiently close to those of p and q, each site of a set gains and loses visibility with each site of the other set around the same time as p and q gained and lost visibility. Since there are $n^2/4$ pairs of sites, this leads to the claimed $\Omega(n^2 m)$ many visibility events. □

Corollary 7 *A set S of n linearly moving sites in a simple polygon of m vertices can have $\Omega(n^2 m)$ geodesic-events.*

References

[1] Imad Afyouni, Cyril Ray, Sergio Ilarri, and Christophe Claramunt. Algorithms for continuous location-dependent and context-aware queries in indoor environments. In *Proc. 20th Int. Conf. Adv. in Geogr. Inf. Sys.*, pages 329–338, 2012.

[2] Boris Aronov, Leonidas J. Guibas, Marek Teichmann, and Li Zhang. Visibility queries and maintenance in simple polygons. *Discrete Comput. Geom.*, 27(4):461–483, 2002.

[3] Julien Basch, Leonidas J. Guibas, and John Hershberger. Data structures for mobile data. *J. Algorithms*, 31(1):1–28, 1999.

[4] Leonidas J. Guibas. Modeling motion. In *Handbook of Discrete and Computational Geometry, Second Edition.*, pages 1117–1134. 2004.

[5] Leonidas J. Guibas and John Hershberger. Optimal shortest path queries in a simple polygon. *J. of Comp. and Syst. Sci.*, 39(2):126–152, 1989.

[6] Menelaos I. Karavelas and Leonidas J. Guibas. Static and kinetic geometric spanners with applications. In *Proc. 12th Ann. ACM-SIAM Symp. Discr. Alg.*, pages 168–176, 2001.

[7] Stéphane Rivière. Dynamic visibility in polygonal scenes with the visibility complex. In *Proc. 13th Ann. Symp. Comput. Geom.*, pages 421–423, 1997.

[8] Yanqiu Wang, Rui Zhang, Chuanfei Xu, Jianzhong Qi, Yu Gu, and Ge Yu. Continuous visible k nearest neighbor query on moving objects. *Information Systems*, 44:1 – 21, 2014.

Competitive Analysis of the Pokémon Go Search Problem

Marc van Kreveld
Utrecht University, m.j.vankreveld@uu.nl

Abstract

Using Pokémon Go as inspiration, we define a search problem on geometric graphs and analyse whether a competitive strategy for it exists. Given a trainer on the graph and a Pokémon anywhere in the plane, the trainer has a sighting of the Pokémon if it is within distance r. She/He can catch it when it is at distance $\leq d$, which is sufficiently smaller than r but of the same order. The problem is to catch the Pokémon once it becomes a sighting quickly, with little detour. We show that for general geometric graphs, there is no constant c such that a c-competitive search strategy for the Pokémon Go search problem exists. On the other hand, if the geometric graph has convex faces or constant geometric dilation, then we can design a c-competitive search strategy for a constant c.

1 Introduction

After its introduction in the summer of 2016, Pokémon Go quickly became the most popular game around. The game requires players—called trainers—to walk outside to search for and catch Pokémon. Now that the first craze is over, it is time for scientists to analyze various aspects of the game. Since the game requires a mobile device with GPS and is (in part) essentially a search problem played on the streets, we can analyze it using competitive analysis. We model the *Pokémon Go search problem* as follows:

- The streets are modeled by a geometric graph G.
- The trainer T is a point on G which can move along the edges and vertices of G, and choose at each vertex any incident edge.
- The Pokémon P to be caught is a point in the plane, not necessarily on G.
- The *sighting range* is a disk with radius r centered on T.
- The *catch range* is a disk with radius d centered on T; we assume that there are two constants c_1, c_2 with $0 < c_1 < c_2 < 1$ such that $c_1 \cdot r < d < c_2 \cdot r$. In words, d is of the same order as r but not nearly r. In Pokémon Go, $r \approx 200\,m$ and $d \approx 30\,m$.
- The GPS of T is continuous and precise, that is, T knows its exact location all the time. Furthermore, T knows G, r, and d, and T knows when P is in the catch range.

The Pokémon Go search problem arises when T gets P as a sighting, so T knows that P is exactly r away. If T would know exactly where P is, then a shortest-path computation on an augmented graph yields the quickest way to catch P (which is to come within distance d from P). The *Pokémon Go search problem* is to get P in the catch range after T gets it in the sighting range in an effective manner, with as little detour as possible. It can be that G has no point that is within distance d from P because P need not be on the graph. Then it is impossible to catch P.

Since we do not know where P is, we can use competitive analysis for a strategy that attempts to find P quickly. A search strategy is *c-competitive* if it guarantees to find P in $c \cdot \tau$ time (or distance) when the quickest possible way to find P takes τ time. The *competitive ratio* c is the worst-case ratio between the time taken by a search strategy and an all-knowing searcher who takes the shortest path.

When the trainer T moves on G and P appears in the sightings, we know that P is somewhere on a half-circle with T in the center. The movement direction up to the moment when P appears determines which half-circle. If P were anywhere on the complementing half-circle, P would have already shown in the sightings of T before.

If there were no streets G, so T could move freely in the plane, then for any $\varepsilon > 0$, a $(1+\varepsilon)$-competitive search strategy exists: Suppose T is at point s when P appears as a sighting. T moves γ further, then moves over a circle with radius γ centered at s. This guarantees that T finds one more position besides s that is exactly at distance r from P. The two half-circles on which P lies have one intersection point in common, and this is where P is. So after moving a distance $\leq (1 + 2\pi)\gamma$, T moves $r - d$ to catch P. The all-knowing searcher simply moves $r - d$ from s towards P. To obtain a $(1 + \varepsilon)$-competitive search strategy, we choose $\gamma = \varepsilon(r - d)/8$.

Related work. We list a few main results on searching and competitive analysis; for a more extensive overview see [4]. Perhaps the best-known search problem is that of finding a point on a line, not knowing how far or to which side from the starting location it

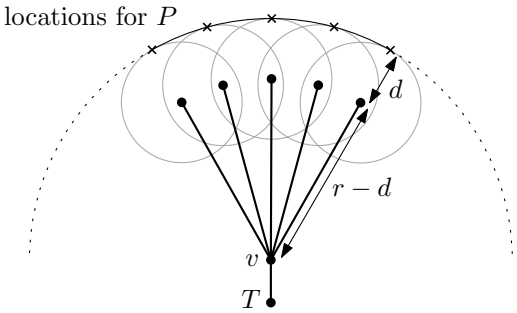


Figure 1: Geometric graph showing that no competitive search strategy can exist.

is. Baeza-Yates et al. [1] showed that a competitive ratio of 9 can be achieved for this 1D problem and this is optimal. Variations with turn cost [3], multiple rays [5], or additional information on the distance [2, 6] were studied as well. Kalyanasundaram and Pruhs [8] considered visibility-based searching for a recognizable point in an unknown scene with convex obstacles. Their result on competitiveness is not constant, but depends on number of obstacles and their aspect ratio. Hoffmann et al. [7] showed that an unknown simple polygon can be discovered completely with a competitive ratio of 26.5. For searching in a known graph for an unknown node, see e.g. [9].

Overview. The remainder of this paper assumes that T can move only on the geometric graph G. For brevity we will often leave out the word "geometric" from "geometric graph". When we mention c-competitive, it is understood that c is constant. When we speak of a competitive strategy, we imply a c-competitive, or constant-factor competitive, strategy.

Section 2 shows that no c-competitive strategy exists for the Pokémon Go search problem that works on all graphs. Section 3 shows that if G has convex faces only, or G has bounded geometric dilation, a c-competitive search strategy exists.

2 General geometric graphs

While we would ideally develop a c-competitive strategy for T to find P on any graph G, we show first that such a strategy does not exist: There are graphs that are not competitively searchable.

Theorem 1 *For any constant $c \geq 1$, a geometric graph exists that has no c-competitive search strategy for the Pokémon Go problem.*

Proof. We construct a graph G and m possible locations of P. The latter lie equally-spaced on a circular arc of radius r which is one-eighth of a circle, see Figure 1. Graph G has a vertex v at the center of the circle supporting this circular arc, so v is at distance r from the possible locations of P. We make m more vertices, each at distance $r - d$ from v and on the line segment from v to a possible location of P. Vertex v is connected to these m vertices, creating a star graph with m "dead ends". v is also connected to one more vertex, away from the possible locations of P. This is where T starts, so when T reaches v, it gets P as a sighting. The graph G can be searched m-competitively but not better, and since m can be any value, there cannot be a c-competitive search strategy for any constant c. □

3 Special classes of geometric graphs

We identify two restrictions on geometric graphs that allow us to prove c-competitiveness. First, we show that graphs with convex faces and a convex cycle bounding the outer face can be searched c-competitively. Then we show that graphs with constant geometric dilation can be searched c-competitively. Notice that the two restrictions are independent: there are graphs with convex faces that do not have bounded geometric dilation and vice versa.

Graphs with convex faces. Let G be a graph with all faces convex. The unbounded face is obviously not convex; instead we assume that it is the complement of a convex polygon. Under these assumptions we can show that a search strategy for G exists that is c-competitive for some constant c.

Our strategy is to find a second location for T that has P on the boundary of the sighting range. This is easy: We let T go clockwise along the boundary of the face to the right. The longest convex path inside the sighting range has length $< 2\pi r$, so within this distance we find this second location. With two locations at distance r from P, only one possible location for P remains, so we can take a shortest path to the catch range. This path has length $\leq 2\pi r + D$, where D is the shortest distance from the first location where T got P in its sighting range, because we can always go back. So the competitive ratio is at most $(4\pi r + D)/D$, and $D \geq r - d$. The ratio is minimized when $D = r - d$. By the assumptions on r and d, our search strategy is c-competitive for a constant c.

This constant can be improved. If T really walks nearly $2\pi r$ to find the second location for the boundary of the sighting range, then T must have gone around the face that has P inside, and T will know this. In this case, P cannot be caught. If T does not go around P and P does not get P into its catch range, then T can move much less far. In any case it is less than $(\pi + 2)r$. We omit a precise analysis in this abstract.

Theorem 2 *Given a geometric graph G with convex bounded faces and a convex outer boundary, two*

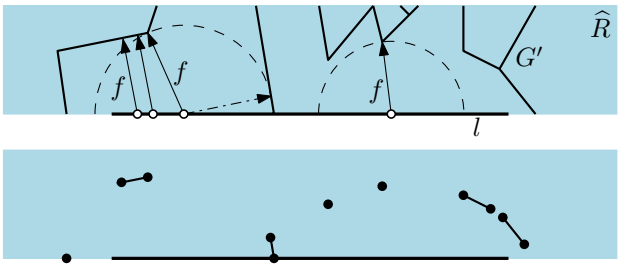


Figure 2: The function f and its image.

constants $0 < c_1 < c_2 < 1$, a sighting range with radius r, a catch range with radius d, and assume $c_1 r < d < c_2 r$, the Pokémon Go search problem can be solved with competitive ratio $O(1)$.

Graphs with bounded geometric dilation. Next we assume that G is a graph whose geometric dilation is bounded by δ. By definition, for any two points p, q on G, the distance between p and q on G is at most δ times the Euclidean distance between p and q.

When T gets P as a sighting, we know that P is at distance r from T and on a specific, known half-circle H. The only locations on G from where P can potentially be caught lie in a half-annulus with circular ends, the Minkowski sum of H with a disk of radius d centered at the origin. Instead of solving the problem directly in this circular setting, we first consider a linearized scenario and then argue that the analysis still holds, albeit with different constants.

Consider a rectangle R with width w and height $d \leq w$. Let l be the lower edge of R, with length w. We assume that R has its lower left vertex at the origin and l lies on the x-axis, so x-coordinates in $[0, w]$ can be used to specify locations on l. Let $\widehat{R}$ be a rectangle of width $w+2d$ and height d; $\widehat{R}$ has a $d \times d$ square added to the left and right of R, see Figure 2. Assume G is a geometric graph and G' is the part inside $\widehat{R}$; edges are clipped and can be partially in G'. In that case we add a new vertex where that edge intersects $\partial\widehat{R}$ so that G' is a proper graph again.

We imagine a disk with radius $\leq d$ whose center is on l and whose boundary intersects G' but its interior does not. For every point on l, such a largest (interior-)free disk is unique. The collection of disks gives rise to a collection of points of G' with contacts with such disks. These contacts are isolated points (vertices) or intervals of edges. Let f be the function that maps each point on l to its closest point on G', see Figure 2. In case of multiple closest points we choose the one most counter-clockwise. The function f has $[0, w]$ as its domain: the x-coordinates that specify points on l.

Lemma 3 *The x-coordinates of the image of f are monotonically increasing.*

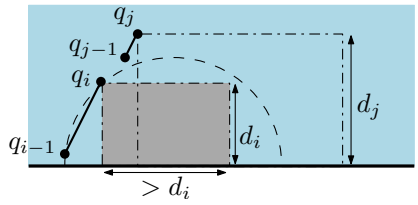


Figure 3: Illustration of the proof of Lemma 5.

Proof. When the contact is an interval, it is clear that it can only move rightward when the largest free disk centered on l moves rightward. When there are two contacts and the function f has a discontinuity, the jump is also always rightward. □

Lemma 4 *The length of the image of f is $O(w)$.*

Proof. Only the contact intervals contribute to the length, not the isolated points. Observe that any interval on an edge is the image of a part of l that is at least as long as that interval. The lemma follows. □

We will bound the summed distance of the jumps in the image of f. As observed in the proof of Lemma 3, every jump is rightward, so every jump is in clockwise direction along the boundary of the largest free disk.

Lemma 5 *The summed distance of all jumps in the image of f is $O(w)$.*

Proof. We first assume that the image of f is a set of points. Let $Q = q_1, \ldots, q_k$ be these points, from left to right. Consider pairs q_{i-1}, q_i of consecutive points and the slope of the line ℓ_i through q_{i-1}, q_i. For all pairs for which this slope is ≥ -2 and ≤ 2, the sum of the jumps is $\leq \sqrt{5}w$ because the sum of the x-jumps is $\leq w$ and the sum of the y-jumps is $\leq 2w$.

Consider all pairs for which the slope of ℓ_i is > 2, and let q_{i-1}, q_i be the leftmost one. Let d_i be the distance of q_i to l. Since the slope of $\ell_i > 2$ and the half-disk whose boundary passes through q_{i-1} and q_i has empty interior, there must be a rectangle R_i of height d_i and width $> d_i$ and with upper-left corner at q_i that contains no points of Q inside, see Figure 3. The jump q_{i-1}, q_i is $< \frac{1}{2}\sqrt{5}d_i$. Let q_{j-1}, q_j be the next leftmost pair for which the slope of ℓ_j is > 2. By the same argument, the distance d_j of q_j to l gives rise to an empty rectangle R_j of height d_j and width $> d_j$.

If q_j is vertically above R_i, then the jump q_{j-1}, q_j is $< \frac{1}{2}\sqrt{5}(d_j - d_i)$, so the jumps q_{i-1}, q_i and q_{j-1}, q_j sum up to $< \frac{1}{2}\sqrt{5}d_j$. If q_j is horizontally to the right of R_i, then we charge the jump q_{i-1}, q_i to progress of distance more than d_i on l.

These arguments together imply that the total summed jump distance of all pairs with slope > 2 is at most $\frac{1}{2}\sqrt{5}w$. By symmetric reasoning this is true

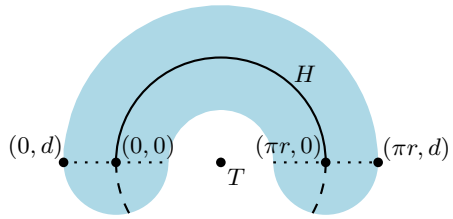


Figure 4: The sausage, the upper and lower sausage, and the coordinate system illustrated.

for slopes < -2. Combining the three cases proves a total jump distance of $2\sqrt{5}w = O(w)$.

If the image of f is a set of points *and intervals on line segments*, the given proof still holds. □

Let us go back to the original problem where we have a half-circle H and the Minkowski sum of H with a disk of radius d, giving a sausage-shape region called S. See Figure 4. We define the outer sausage as the part of S outside the supporting circle of H; the inner sausage is the other part. We need to analyze distances where H has the role of l and the outer (inner) sausage has the role of $\widehat{R}$. The situation is not symmetric for the outer and inner sausage, but the same properties hold.

Instead of left-to-right in R, we use the angle with respect to T, the center of the (half-)circle H. We let the ccw-most point on the half-circle have first coordinate 0 and the cw-most point have first coordinate πr. Other points in the outer or inner sausage have as their first coordinate the same as the point on H with the same angle from the circle center. The second coordinate is the distance to T minus r, so it is zero on H. In this coordinate system, the distance between any two points in the sausage is at most a constant factor off their distance in the Euclidean setting; the precise constant depends on d and r. We can again imagine a maximal interior-free disk with its center on H and its contacts with G inside the outer or inner sausage. These contacts are monotone in the first coordinate. We can show that all previous properties and proofs still hold, but with different constant factors. The added half-disk caps do not influence the asymptotic bounds.

The competitive strategy for T to find P is as follows. Globally, T goes from the starting location (the center of the circle supporting H) to the ccw-most contact in the inner sausage using a shortest path in G. Then T works its way along the contacts in the inner sausage in clockwise direction. When a jump occurs in the contacts, T takes the shortest path in G. When the cw-most point is reached, T traverses the contacts in the outer sausage in the same way, but in ccw order. To go from the last contact in the inner sausage to the first contact in the outer sausage, we use the shortest path in G.

We observe that if P can be found, it will be found if we visit all contacts because these are the points on G closest to the possible locations of P. So the search strategy is correct. Let $\delta \geq 1$ be the geometric dilation of G. For the move from the starting location to the first point in the inner sausage, T traverses a distance at most $r\delta$ on G. All jumps in the inner and outer sausage together require a distance $O(r\delta)$, and the same is true for switching from the inner to the outer sausage. So the traversed distance by this strategy is $O(r\delta)$, whereas the optimal strategy cannot use less distance than $r - d$. The assumptions on d and r imply a competitive ratio of $O(\delta)$.

Theorem 6 *Given a geometric graph G with geometric dilation δ, two constants $0 < c_1 < c_2 < 1$, a sighting range with radius r, a catch range with radius d, and assume $c_1 r < d < c_2 r$, the Pokémon Go search problem can be solved with competitive ratio $O(\delta)$.*

References

[1] Ricardo A. Baeza-Yates, Joseph C. Culberson, and Gregory J. E. Rawlins. Searching in the plane. *Inf. Comput.*, 106(2):234–252, 1993.

[2] Prosenjit Bose, Jean-Lou De Carufel, and Stephane Durocher. Searching on a line: A complete characterization of the optimal solution. *Theor. Comput. Sci.*, 569:24–42, 2015.

[3] Erik D. Demaine, Sándor P. Fekete, and Shmuel Gal. Online searching with turn cost. *Theor. Comput. Sci.*, 361(2-3):342–355, 2006.

[4] Subir Kumar Ghosh and Rolf Klein. Online algorithms for searching and exploration in the plane. *Computer Science Review*, 4(4):189–201, 2010.

[5] Mikael Hammar, Bengt J. Nilsson, and Sven Schuierer. Parallel searching on m rays. *Comput. Geom.*, 18(3):125–139, 2001.

[6] Christoph A. Hipke, Christian Icking, Rolf Klein, and Elmar Langetepe. How to find a point on a line within a fixed distance. *Discrete Applied Mathematics*, 93(1):67–73, 1999.

[7] Frank Hoffmann, Christian Icking, Rolf Klein, and Klaus Kriegel. The polygon exploration problem. *SIAM J. Comput.*, 31(2):577–600, 2001.

[8] Bala Kalyanasundaram and Kirk Pruhs. A competitive analysis of algorithms for searching unknown scenes. *Comput. Geom.*, 3:139–155, 1993.

[9] Elias Koutsoupias, Christos H. Papadimitriou, and Mihalis Yannakakis. Searching a fixed graph. In *Proc. 23rd ICALP*, volume 1099 of *LNCS*, pages 280–289. Springer, 1996.

Computational complexity and bounds for Norinori and LITS

Michael Biro * Christiane Schmidt †

Abstract

Norinori (aka Dominnocuous) and LITS (aka Nuruomino) are pencil-and-paper puzzles played on $m \times n$ square grids. In this paper we show that both Norinori and LITS are NP-complete and that their associated counting problems are #P-complete. Furthermore, we display $m \times n$ boards for each game that have unique solutions using the minimal number of polyomino regions.

1 Introduction

Norinori and *LITS* are a pair of related pencil-and-paper puzzles, made popular by the Japanese publisher Nikoli [3, 4]. The games are each played on an $m \times n$ square grid that has been partitioned into connected polyomino regions. To solve each puzzle, the player is required to place black squares in the polyomino regions to satisfy certain conditions.

In Norinori, the solver places black squares in the polyominos such that the final board satisfies the following two properties:

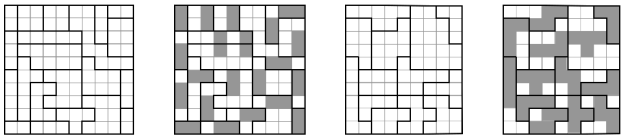
1. Each black square has exactly one black neighbor.
2. There are exactly 2 black squares in each polyomino region.

See Figure 1(a)/(b) for an example Norinori board and its solution. When discussing placing black squares, it is often useful to think of the player as placing black dominos as a basic move, with no two dominos adjacent, see Figure 2(a). A domino may span two polyomino regions, see Figure 1(b).

In LITS, the solver places black squares in the regions such that the final board satisfies the following properties:

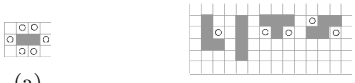
1. The black squares form a connected polyomino.
2. Each polyomino region contains a connected black tetromino.
3. No two congruent tetrominos are adjacent.
4. Black squares may not build 2×2 squares.

See Figure 1(c)/(d) for an example LITS board and its solution. For LITS, it is useful to think of the player as placing one of the four legal black tetrominos, see Figure 2(b), which resemble the letters in



(a) (b) (c) (d)

Figure 1: Example Norinori board (a) and LITS board (c) with solution in (b) and (d), respectively.



(a) (b)

Figure 2: Basic shapes for Norinori (a) and LITS (b), circles give white squares that may not be filled in.

LITS. A variant of LITS, without the condition that no two congruent tetrominos are adjacent, was considered by McPhail [2] and shown to be NP-complete.

In this paper we will show that the problem of solving Norinori and LITS boards is NP-complete and that the problem of counting the number of solutions to a Norinori or LITS board is $\#P$-complete. For our reductions, we will use the PLANAR 1-IN-3-SAT PROBLEM, a well known NP-complete and $\#P$-complete problem [1].

Definition 1 *An instance F of the PLANAR 1-IN-3-SAT problem is a Boolean formula in 3-CNF consisting of a set $\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$ of m clauses over n variables $\mathcal{V} = \{x_1, x_2, \ldots, x_n\}$. Clauses in F contain variables and negated variables, denoted as* literals. *A clause is satisfied if and only if it contains exactly one* `true` *literal, and the formula F is true if and only if all its clauses are satisfied. The variable-clause incidence graph G is planar and it is sufficient to consider formulae where G has a rectilinear embedding.*

In addition, we explore combinatorial questions on both puzzles: the smallest number of regions in an $n \times m$ board that has a unique solution. We show that for most boards only 3 regions are required.

2 NP-completeness of Norinori

Theorem 1 *Determining if a Norinori board is solvable is NP-complete and counting the number of solutions is #P-complete.*

Proof. The proof is by reduction from PLANAR 1-IN-3-SAT. Given an instance F of planar 1-in-3-

*Department of Mathematics and Statistics, Swarthmore College, mbiro1@swarthmore.edu.

†Communications and Transport Systems, ITN, Linköping University, christiane.schmidt@liu.se.

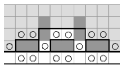
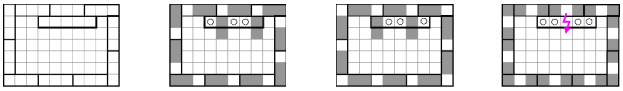
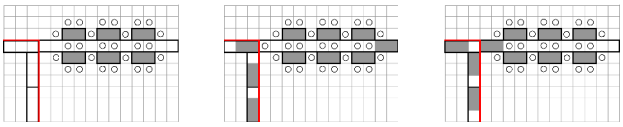


Figure 3: Gadget to fix the 2 filled-in squares in each whitespace (here, for clarity, indicated in light gray): at one corridor we add the region in U-shape.



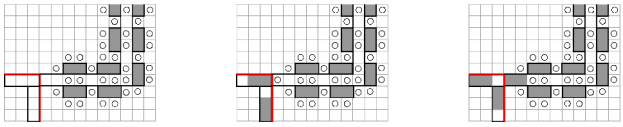
(a) (b) (c) (d)

Figure 4: (a) Variable loop, with the two feasible solutions (b)/(c). We associate the solution in (b) and (c) with a truth setting of "false" and "true", respectively. The 2×5 region in the center face ensures that for (b) and (c) the filled-in squares are fixed, and it renders the third solution for the loop of the 1×3 rectangles infeasible (d).



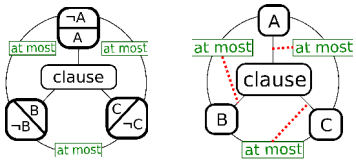
(a) (b) (c)

Figure 5: (a) Corridor gadget with enforced white/black pixels. The connected variable gadget is located within the red boundary. If the variable is set to "false", only a 2×1-block pushed to the end of the corridor is a feasible fill-in (b). If the variable is set to "true", it is possible to place a 2×1-block directly at the connection to the variable loop, leaving the last pixel of the corridor white (c).



(a) (b) (c)

Figure 6: (a) The bend gadget with enforced white and black pixels. The connected variable gadget is located within the red boundary. (b) "false", (c) "true'.



(a) (b)

Figure 7: 1-in-3 gadget construction for Norinori (a) and LITS (b). The red, dotted lines indicate the connectors from the "at most"-gadget.

(a) (b) (c) (d)

Figure 8: (a) The at-most gadget: corridors from two negated variables enter. If both corridors enter with a setting of "true" (variable setting of "false") (b), or if the corridors have different truth settings (c), there exists a feasible solution. (d) If both corridors enter with a variable setting of "true" the board cannot be completed.

SAT with incidence graph G, we show how to turn a rectilinear planar embedding of G into a Norinori board B such that a solution to B yields a solution to F, thereby showing NP-completeness. Furthermore, there will be a one-to-one correspondence between solutions of B and solutions of F, showing #P-completeness.

We begin by constructing a representation of variables and their negations, then show how to propagate and bend the variable values using 'wires' and combine the wires to form clauses. Note that throughout, the constructed gadgets have a single solution for any given variable assignment, which will make this reduction parsimonious.

The polyomino regions not incorporated into our gadgets will not affect the gadget solutions, as for any open region we use a **face gadget**, shown in Figure 3, to force the region to contain two black squares, which disallows any others. Therefore, there can be no interference between our gadgets and the polyomino regions surrounding them.

The basic variable gadget is created out of 1×3 polyominos, and forms a **variable loop**, shown in Figure 4. Each variable loop has two possible solutions, corresponding to setting the variable as "true" or "false", and the domino placement is completely determined by the variable assignment. (A third solution placing the dominoes over the region boundaries

From each variable loop, we can propagate the variable value, creating a **corridor gadget**, shown in Figure 5. Note that a "false" assignment for a variable forces a domino to be placed at the far end of the corridor, while a "true" assignment leaves open the possiblity of placing a domino at either end. This will be accounted for in the final clause gadget.

By choosing the appropriate place to connect the corridor gadget to the variable loop, we can generate wires for both the variable value and its negation. That is, no separate negation gadget is needed. In addition, we can connect several corridor gadgets to the variable loop. Furthermore, we can create 90° turns in the corridor gadget using the **bend gadget**, shown in Figure 6.

To combine the corridor gadgets into clauses, we use the **1-in-3 gadget** in Figure 7(a), where the **at-most** and the **clause gadget** is shown in Figure 8 and 9, respectively. The 1-in-3 gadget uses two negated copies of each variable assignment and combines them with the at-most gadgets. This forces at most one of the variables' truth assignments to be true, while the center clause gadget requires at least one true assignment. Together, this forces exactly 1 true assignment, giving a solution to the instance F.

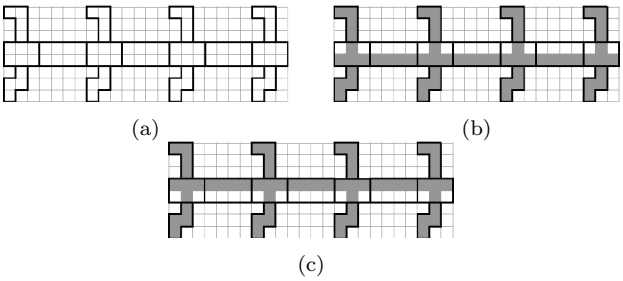
Given a solution to an $m \times n$ Norinori board, it can obviously be verified in polynomial time. □

(a) (b) (c) (d)

Figure 9: Norinori clause gadget (a): if all variables do not fulfill the clause, the clause region cannot be filled (b). (c), (d): feasible solutions for three and one variable fulfilling the clause.

(a) (b)

Figure 10: Gadget for each face of the arrangement: at one corridor of the whitespace (here, for clarity, indicated in light gray) we add a T in distance 4 from an S (a), this enforces the placement of an I to connect the T.



(a) (b) (c)

Figure 11: Variable gadget (a), with the two possible feasible solutions (b),(c). We associate one with a truth setting of "false" (b) and the other with "true" (c).

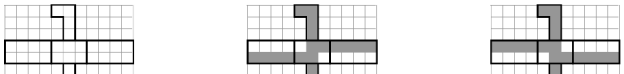
3 NP-completeness of LITS

Theorem 2 *Determining if a LITS board is solvable is NP-complete and counting the number of solutions is #P-complete.*

Proof. The proof is again by reduction from PLANAR 1-IN-3-SAT. The structure of the LITS reduction is the same as in the previous proof for Norinori. The properties of a final LITS board enforce unique feasible solutions for the following gadgets.

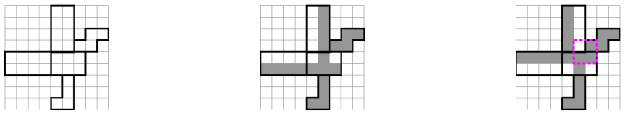
We begin by noting that the polyomino regions not incorporated into our gadgets will not affect the gadget solutions, as for any open region we use a **face gadget**, shown in Figure 10, to force the region to contain a connecting 4×1 tetromino, which disallows any others. Therefore, there can be no interference between our gadgets and the polyomino regions surrounding them.

The basic **variable gadget** is created out of a repeating pattern of polyominos, shown in Figure 11. Each variable gadget has two possible solutions, corresponding to setting the variable as "true" or "false", which are completely determined by the tetromino placement. The variables are connected by 4×1 tetrominos, similar to the face gadget. This ensures that the resulting set of black squares can be connected.



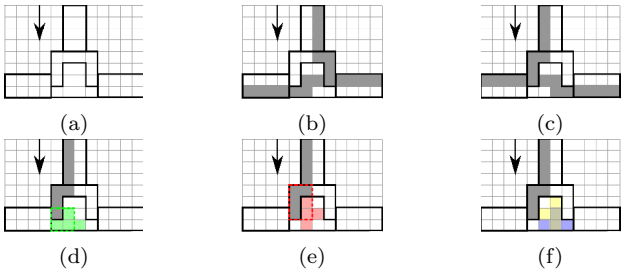
(a) (b) (c)

Figure 12: (a) The NOT gadget. (b),(c) The wires connected by the NOT gadget always satisfy opposite truth assignments.



(a) (b) (c)

Figure 13: (a) The bend gadget. As the 3x2 rectangle is adjacent to an enforced L, S and I (length of the 2x4 rectangle), it must be filled in with a T. In (b) the other T would not connect to the incoming I; the other outgoing I would leave the S unconnected. In (c) the other T would not connect to the incoming I; the other outgoing I would result in a filled 2x2 square shown in dashed pink.



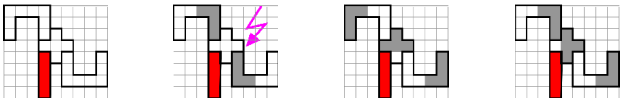
(a) (b) (c) (d) (e) (f)

Figure 14: (a) The split gadget, with the two different truth settings (b), (c). The central shape (adjacent to an L and an I) must contain either a S or a T. No position of the T is possible (d)-(f).

We can propagate the variable assignment, creating a **corridor gadget**, by linearly repeating the pattern in the variable gadget. Negating a variable corresponds to inserting a **NOT gadget** into the corridor, as in Figure 12. To create 90° turns in the corridor gadget, we use the **bend gadget**, shown in Figure 13, and to create copies of the variable assignment we use the **split gadget**, shown in Figure 14.

To combine the corridor gadgets into clauses, we use the **1-in-3 gadget** in Figure 7(b), where the **at-most** and the **clause gadget** is shown in Figure 15 and 16, respectively. The 1-in-3 gadget combines two copies of each variable's assignment with the others using the at-most gadgets. This forces at most one of the variables' truth assignments to be true, while retaining the connectivity condition. Then, the center clause requires at least one true assignment and together these force exactly 1 true assignment, giving a solution to the instance F.

Given a solution to an $m \times n$ LITS board, it can obviously be verified in polynomial time. □



(a) (b) (c) (d)

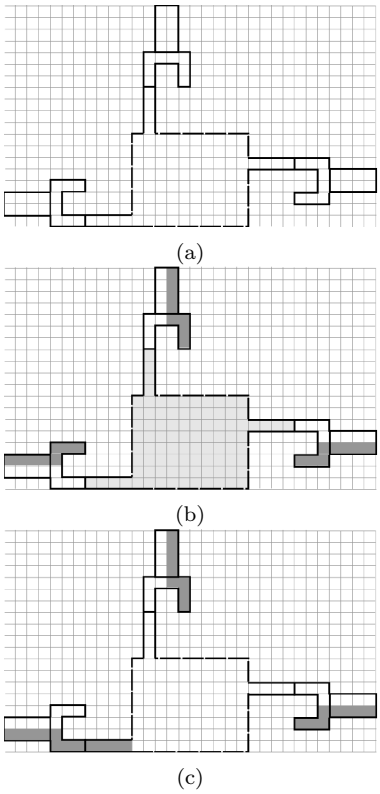
Figure 15: (a) At-most gadget: the two C-shaped regions connect to the variable corridors (as in the clause gadget); the red I is a connector (corridor of (enforced) I- and T-shapes that connects to an S or L of a corridor on that face), as indicated by the red lines in Fig. 7(b). If both variables have a truth setting fulfilling the clause (b), no T in the central cross can be placed without filling a 2x2-square. If both variables do not fulfill the clause (c), or one of them does (d), a T can be placed.

4 Boards with unique solutions

Definition 2 *Define $U_N(n,m)$ to be the minimal number of regions among all $n \times m$ Norinori boards with unique solutions. Similarly, define $U_L(n,m)$ to be the minimal number of regions among all $n \times m$ LITS boards with unique solutions. Note that $U_N(n,m)$ and $U_L(n,m)$ need not exist for all n,m, in which case we say $U_N(n,m) = 0$ or $U_L(n,m) = 0$.*

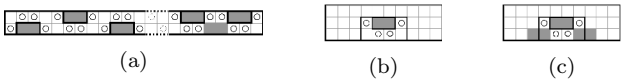
Theorem 3 *The following values for $U_N(n,m)$ hold:*

1. $U_N(n,1) = 0$ *for* $n \not\equiv 2 \mod 3$
2. $U_N(n,1) = \frac{n+1}{3}$ *for* $n \equiv 2 \mod 3$
3. $U_N(n,2) \leq \lceil \frac{n}{4} \rceil$ *for* $n \geq 3$
4. $U_N(n,m) = 3$ *for all* $n \geq 5, m \geq 3$.



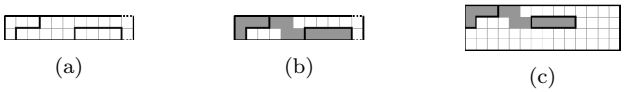
(a) (b) (c)

Figure 16: (a) The clause gadget: if all variables have truth settings that do not fulfill the clause (b), no tetromino in the light gray region can be connected; if at least one variable has a truth setting fulfilling the clause (c), an I can connect to the other tetrominoes.



(a) (b) (c)

Figure 17: (a) For $m = 2$, $\lceil \frac{n}{4} \rceil$ regions can have a unique solution. (b)/(c) For $m \geq 3$ and $n \geq 5$ there exist 3 polyomino regions, such that the board has a unique solution.



(a) (b) (c)

Figure 18: For $m \geq 2$ and $n \geq 10$ there exist 3 polyomino regions (a), such that the board has a unique solution (b). An example for m=4, n=14 (c).

Proof. For the case of an $n \times 1$ board, examine the leftmost domino in the completed board. Since the board has a unique solution, it must consist of the leftmost two squares, as otherwise we could move it left without making the board infeasible. Furthermore, the leftmost region can consist of either 2 or 3 squares, as otherwise we could move the domino right without making the board infeasible. Therefore, we generate an $(n-3)\times 1$ board by removing the leftmost three squares and, if necessary, modifying the second leftmost region by removing its leftmost square. The resulting board has the same unique solution as the original, restricted to its squares. We can then repeat the process until there is only one region and verify that the only $n \times 1$ board with one region that has a unique solution is the 2×1 board. Therefore, the only $n \times 1$ boards with a unique solution have $n \equiv 2 \mod 3$. Moreover, we remove one domino per three squares, so $U_N(n,1) = \frac{n+1}{3}$ in this situation.

For the other cases, see Figure 17 for constructions achieving the given bounds, with the observation that $U_N(n,m) > 2$ for $n \geq 5, m \geq 3$. □

Theorem 4 *$U_L(n,m) = 3$ for all $n \geq 10, m \geq 2$. In other words, 3 regions suffice to completely determine an $n \times m$ LITS board, as long as $n \geq 10$ and $m \geq 2$.*

Proof. See Figure 18. □

References

[1] M. E. Dyer and A. M. Frieze. Planar 3DM is NP-complete. *Journal of Algorithms*, 7(2):174–184, 1986.

[2] B. McPhail. Metapuzzles: Reducing SAT to your favorite puzzle. *CS Theory Talk*, 2007.

[3] Nikoli. http://www.nikoli.com/en/puzzles/norinori/, NIKOLI Co., Ltd. Accessed December 6, 2016.

[4] Nikoli. http://www.nikoli.co.jp/en/puzzles/lits.html, NIKOLI Co., Ltd. Accessed December 6, 2016.

How to play hot and cold on a line

Herman Haverkort* David Kübel† Elmar Langetepe† Barbara Schwarzwald†

1 Introduction

Imagine we want to set up receivers to locate an animal that carries a device which sends signals. The strength of the signals may vary, but we know one thing: the further the distance from the animal, the weaker the signal. Or imagine a researcher conducting a survey, who wants to summarize respondents' political preferences by scoring them on several scales (for example, from conservative to progressive, or from favouring a small state to a large state). Respondents may not be able to score themselves but, given a number of hypothetical party programmes, they can rank them and say which one they like best.

In such settings, we are essentially searching for a target that is a point in a one- or higher-dimensional space. To pinpoint the location of the target, we issue queries (receivers, party programmes) that are points in the same configuration space. As a response, we obtain an ordering of the query points according to their distance to the target. From this we try to derive the location of the target with the highest possible accuracy—or conversely, we try to reach high accuracy with as few (expensive) queries as possible.

Searching for a stationary target is a common problem in computer science and applied mathematics. Strategies such as evenly distributed query points or binary search spring to mind immediately, but, as we will see in this paper, at least in certain abstract settings of the problem we can do much better. The problem of efficiently obtaining an order on a set of objects has been studied before in very general settings [3], but note that in our case, the cost measure is the number of query points that are used, not the number of comparisons that are made between them. The reconstruction of geometric objects based on a sequence of geometric probes (points, lines, hyperplanes, wedges, etc.) has also been investigated: the problem was introduced by Cole and Yap [2] and the main focus is also on the number of queries [5].

Specifically, we focus on the following setting. The target t is a one-dimensional point located at an unknown position in the unit interval $[0, 1]$. To pinpoint the location of t, we may query the interval at points $q_1, \ldots, q_n \in [0, 1]$. As a response, we obtain an ordering of the points by ascending distance to t. This restricts possible positions of t to a subinterval bounded by bisectors of query points or an end point of the initial interval. We measure the efficiency or quality of a query strategy in terms of the reciprocal of the size of the subinterval in which the target t is found to lie. The worst-case of this reciprocal, that is, the minimum over all possible locations of t, is called the *accuracy* of the query strategy.

With respect to the frequency of the responses, we distinguish two variants. In the *one-shot* variant (Sec. 2), the response is only given after all points $q_1, \ldots q_n$ have been placed. In this case one needs to maximize the number of different, well-spaced bisectors. Such combinatorial questions are classical problems in discrete geometry; see for example [4]. In the *incremental* variant (Sec. 3), a response is given after each point placement, and may affect the choice of the next point. This enables a binary search strategy, but we can do much better than that. The problem can be interpreted as a game where an adversary tries to hide the target in the largest possible area. Geometric games about area optimization have a tradition in Computational Geometry; see for example [1].

For both variants we present an efficient strategy and an upper bound on the accuracy that can be achieved. In Section 4, we briefly discuss room for improvement and implications for higher-dimensional settings of the problem.

2 One-shot strategies

First we consider the one-shot variant of the problem: only after generating n query points, we get to hear their ordering by distance to the target t. This pinpoints t to a subinterval bounded by bisectors of query points or an end point of the interval. As the target may lie in any subinterval, our problem is equivalent to minimizing the maximum size of such an interval.

As n query points can produce at most $\frac{1}{2}n(n-1)$ distinct bisectors, there are at most $\frac{1}{2}n(n-1)+1$ intervals. Thus, we get the following (trivial) upper bound for the accuracy of one-shot strategies:

Theorem 1 *The accuracy of any one-shot strategy with n points is at most $\frac{1}{2} \cdot (n^2 - n + 2) \in O\left(n^2\right)$.*

We will now develop a strategy to get close to this upper bound. As a starting point, consider the following simple strategy, which we call EquiDist(n): place n evenly spaced query points $(q_1, q_2, \ldots, q_i, \ldots, q_n) := (0, \frac{1}{n-1}, \ldots, \frac{i-1}{n-1}, \ldots, 1)$. The accuracy of EquiDist(n) is only $2(n-1)$ because

*Dept. of Math. and Computer Sc., TU Eindhoven
†Dept. of Computer Sc., U. of Bonn.

many bisectors overlap. However, for $n \geq 7$ it is possible to forgo some of these query points, while the number of distinct bisectors, and so the accuracy, stays the same. Hence, for some function $\varphi(n)$ with $\varphi(n) > n$, we can achieve the same accuracy as EquiDist($\varphi(n)$) with n query points. Specifically, we now introduce a strategy Gap$_x(n)$ that will form a subset of the query points defined by EquiDist($\varphi_x(n)$) for $\varphi_x(n) := n(x+1) - 2x^2 - x - 2$ by using only the following n points from EquiDist($\varphi_x(n)$): $q_1, q_2, \ldots, q_{x+1}$, as well as $q_{2x+1+k\cdot(x+1)}$ for $k \in \{0, \ldots, n-(2x+3)\}$, and $q_{\varphi_x(n)-x}, \ldots, q_{\varphi_x(n)-1}, q_{\varphi_x(n)}$. This results in widely spaced query points throughout the search range, and tightly spaced points near both ends, omitting at most x consecutive query points from EquiDist($\varphi_x(n)$). Most bisectors will then be formed by one of the tightly and one of the widely spaced points.

Lemma 2 *For $x, n \in \mathbb{N}$ with $n \geq (2x+3)$, the one-shot query strategy* Gap$_x$ *has accuracy $2(\varphi_x(n) - 1)$.*

Proof. It suffices to show that every distinct bisector from EquiDist($\varphi_x(n)$) is also created by Gap$_x(n)$, as the points chosen by Gap$_x(n)$ are a subset of those chosen by EquiDist($\varphi_x(n)$). For each $1 \leq i < \varphi_x(n)$, Gap$_x(n)$ must choose two points that form a bisector in the middle between q_i and q_{i+1}. For each $1 < j < \varphi_x(n)$, Gap$_x(n)$ must choose two points that form a bisector directly on q_j.

For $i \leq x$ and $j \leq x$ this is given by $q_1, q_2, \ldots, q_{x+1}$. For $x < i < \frac{\varphi_x(n)}{2}$, EquiDist($\varphi_x(n)$) forms a bisector between q_i and q_{i+1} with all of the pairs $(q_1, q_{2i}), (q_2, q_{2i-1}), \ldots, (q_{x+1}, q_{2i-x})$. One of these pairs has to be in the chosen subset of Gap$_x(n)$, since the choosing process omits at most x consecutive points. Likewise, one of the pairs (q_1, q_{2j-1}), $(q_2, q_{2j-2}), \ldots, (q_{x+1}, q_{2j-(x+1)})$ must have been chosen by Gap$_x(n)$. Those pairs form a bisector on q_j for $x < j \leq \frac{\varphi_x(n)}{2}$. The existence of the remaining bisectors, that is, those in the right half of the unit interval, follows by the symmetry of the strategy.

Hence Gap$_x(n)$ produces the same set of $2(\varphi_x(n) - 2)$ distinct equidistant bisectors as EquiDist($\varphi_x(n)$), resulting in an accuracy of exactly $2(\varphi_x(n) - 1)$. □

It remains to choose the optimal x, given n, maximizing $\varphi_x(n)$. As the slope of φ_x increases with x, Gap$_x$ is eventually surpassed by Gap$_{x+1}$. The break-even point between Gap$_x$ and Gap$_{x+1}$ is at $n = 4x + 3$. Hence, $x_{opt} = \lceil \frac{n-3}{4} \rceil$ and we get:

Theorem 3 *For any $n \geq 3$ and $x_{opt} = \lceil \frac{n-3}{4} \rceil$, the one-shot strategy* Gap$_{x_{opt}}(n)$ *has an accuracy of $2(\varphi_{x_{opt}}(n) - 1) \geq \frac{1}{4} \cdot (n^2 + 6n - 27) \in \Omega(n^2)$.*

This leaves only a gap of a factor two between the lower and the upper bound.

3 Incremental and online strategies

In this section we consider incremental strategies, that is, before placing any query point q_i we may learn the ordering of $q_1, \ldots, q_{i-1}$ by (increasing) distance to the target, and we may choose the location of q_i depending on that information. We will show upper and lower bounds on the accuracy that can be achieved.

3.1 A strategy with high accuracy

A simple incremental strategy could choose the query points such that the interval containing the target is halved in each step. Thus, with n query points, we achieve accuracy 2^{n-1}. But we can do much better, with a recursive strategy that takes advantage of more than one new bisector in many steps. Let $h(n)$ be the accuracy we aim for, as a function of the number n of query points we get to place. For ease of description, we use the interval $[0, h(n)]$ instead of the unit interval and pinpoint the target to an interval of length 1. The recursion in our strategy depends on a number of conditions, labelled A to E and presented below.

Our strategy places the first two query points symmetrically. Let $g(n)$ be the distance of the first two query points to their common bisector, so we place q_1 at $\frac{1}{2}h(n) - g(n)$ and q_2 at $\frac{1}{2}h(n) + g(n)$. Now suppose the target lies to the right of the bisector (the other case is symmetric). We now place q_3 at some distance $2f(n)$ to the right of q_2, and find that the target lies in one of three intervals (see Figure 1(i)):

(i) $[\mathrm{bs}(q_1, q_2), \mathrm{bs}(q_1, q_3)] = [\frac{1}{2}h(n), \frac{1}{2}h(n) + f(n)]$, of size $f(n)$.

(ii) $[\mathrm{bs}(q_1, q_3), \mathrm{bs}(q_2, q_3)] = [\frac{1}{2}h(n) + f(n), \frac{1}{2}h(n) + g(n) + f(n)]$, of size $g(n)$, or

(iii) $[\mathrm{bs}(q_2, q_3), h(n)] = [\frac{1}{2}h(n) + g(n) + f(n), h(n)]$, of size $\frac{1}{2}h(n) - g(n) - f(n)$.

To be able to apply our strategy recursively in the first interval, using the remaining $n-3$ query points, we choose $f(n) = h(n-3)$. For the third interval, of width $\frac{1}{2}h(n) - g(n) - h(n-3)$, we use an adapted strategy, explained below, that places the remaining $n-3$ query points in a way that exploits the previously placed query points q_2 and q_3 at distance $f(n) = h(n-3)$ left and right of its left boundary. The same strategy can be applied symmetrically to the second interval, provided the second interval is not larger than the third, that is, $g(n) \leq \frac{1}{4}h(n) - \frac{1}{2}h(n-3)$ for $n \geq 3$ (condition **A**). Note that condition A also ensures that q_1 and q_2 lie within the interval $[0, h(n)]$. To be able to place q_3, we also require $g(n) + 2h(n-3) \leq \frac{1}{2}h(n)$ for $n \geq 3$ (condition **B**).

We now describe our adapted strategy to locate a target in an interval $[0, \frac{1}{2}h(n) - g(n) - h(n-3)]$ (modulo translation) with $n-3$ query points $q_4, \ldots, q_n$ that can be chosen freely and two predetermined query points $q_2 = -h(n-3)$ and $q_3 = h(n-3)$ (see Figure 1(ii)). We place q_4 at $h(n-3) + 2h(n-4)$; this is possible if $h(n-3) + 2h(n-4) \leq \frac{1}{2}h(n) - g(n) - h(n-3)$

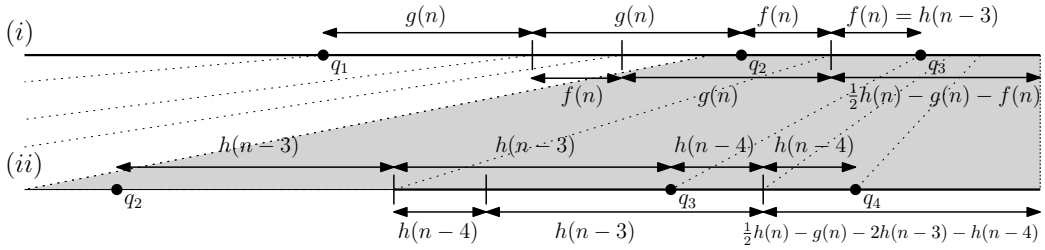


Figure 1: Location of the first query points for our incremental strategy to locate a target with accuracy $h(n)$.

for $n \geq 4$ (condition **C**). Again, the target lies in one of three intervals: an interval of width $h(n-4)$ on the left, to which we apply the overall strategy recursively, an interval of width $\frac{1}{2}h(n)-g(n)-2h(n-3)-h(n-4)$ on the right, with predetermined query points at distance $h(n-4)$ from its left boundary, and an interval of width $h(n-3)$ in the middle, with predetermined query points at distance $h(n-4)$ from its right boundary. We can apply the adapted strategy recursively to the rightmost interval, using the remaining $n-4$ query points, provided $\frac{1}{2}h(n)-g(n)-2h(n-3)-h(n-4) = \frac{1}{2}h(n-1)-g(n-1)-h(n-4)$ for $n \geq 4$ (condition **D**), and we can apply the adapted strategy recursively to the interval in the middle if $h(n-3) \leq \frac{1}{2}h(n-1)-g(n-1)-h(n-4)$ for $n \geq 4$ (condition **E**). nn It remains to choose h and g as functions of n such that the conditions **A**–**E** for recursion are satisfied, and such that, when no further recursion is possible because we have run out of query points, the remaining interval is small enough: $h(0) \leq 1$, $h(1) \leq 1$, $h(2) \leq 2$, and $\frac{1}{2}h(3)-g(3)-h(0) \leq 1$. The optimal choice of h and g that satisfies these conditions turns out to be $h(0) = h(1) = 1$, $h(2) = 2$, $h(3) = 6$, $g(n-1) = \frac{1}{4}h(n-1)-\frac{1}{2}h(n-4)$ for all $n \geq 4$ (satisfying condition A with equality), and (substituting this in condition D) $h(n) = h(n-1)+6h(n-3)+2h(n-4)$ for all $n \geq 4$. Thus $h(n) = \Omega(b^n)$, where $b > 2.2993$ is the largest root of $b^4 - b^3 - 6b - 2$, and we obtain:

Theorem 4 *There is an incremental strategy to locate a target in a unit interval with accuracy $\Omega(b^n)$, where $b > 2.2993$ is the largest root of b^4-b^3-6b-2.*

Theorem 4 also holds in the on-line setting, where n is unknown until the last query point has been placed. In the on-line setting, the strategy is to take $h(n) = 2b^{n-2}$ and $g(n) = \frac{1}{4}h(n)-\frac{1}{2}h(n-3) = (\frac{1}{2}-b^{-3})b^{n-2}$. Thus, the locations of $q_1, ..., q_4$, relative to the size of the interval, $h(n)$, are independent of n, and can therefore be decided without advance knowledge of n.

3.2 Upper bound

By placing a query point q_i, we obtain $i-1$ new bisectors of q_i with previously placed query points. We learn the rank of q_i among $q_1, ..., q_i$ with respect to the distance to the target, which tells us where the target is with respect to the new bisectors. This reduces the interval R where the target must be to one of at most i subintervals of R, at least one of which must have size at least $1/i$ times the size of R. Therefore, for any incremental strategy, there must be targets for which we can increase the accuracy with a factor at most i with every query point q_i, leading to an accuracy of at most $n!$ after placing n query points. However, this bound is far from tight. We can show a much stronger upper bound on the accuracy that can be achieved:

Theorem 5 *For any deterministic incremental strategy on the unit interval, there is a target that cannot be located with accuracy better than $O(b^n)$, where $b < 3.652$ is the greatest root of $b^3 - 4b^2 + b + 1$.*

Let R_n be the interval in which a target is known to lie after placing n query points, let r_n be the size of R_n, and let a_n be the corresponding accuracy $1/r_n$. Our proof of Theorem 5 is based on the following insight. Suppose, for example, that we have placed $n-1$ query points, and placing another query point q_n generates two new bisectors that divide the target interval into three equal parts—thus improving the accuracy by a factor three. This means that q_n generates two new bisectors $\mathrm{bs}(q_n, q_i)$ and $\mathrm{bs}(q_n, q_j)$, where q_i and q_j are two previously placed query points that are consecutive in the left-to-right order of all previously placed points, and such that the distance between q_i and q_j is 2/3 of r_{n-1} (the size of the target region after $n-1$ query points) and two times r_n (the size of the target region after n points).

The example illustrates that, for a large increase in accuracy, one needs to have *close pairs*: pairs of previously placed query points with a small distance between them, compared to the size of the current region. The example also illustrates that, as the accuracy increases, close pairs can stop being close as their distance relative to the current accuracy increases. Moreover, with the placement of each query point, one can create at most two new close pairs (one with the left neighbour and one with the right neighbour among the query points placed so far). To create a close pair, one must place a query point q_n close to a previously placed query point, but then the newly

created bisectors are also close to the previously created bisectors. In particular, if a close pair is created, most new bisectors inside R_{n-1} will be relatively close to the boundary of R_{n-1} and away from the centre of R_{n-1}. Thus, creating close pairs cannot go together with substantial accuracy gains if the target is near the centre of R_{n-1}. In effect, close pairs are a resource whose scarcity limits the accuracy that can be obtained. Below we make this insight more precise.

We define a *d-close pair* as a pair of previously placed query points that are consecutive in their left-to-right order and with distance less than d divided by the current accuracy. The following three lemmas (proof omitted in this abstract) capture the trade-off between, on the one hand, creating close pairs, and on the other hand, using and losing them while pinpointing the location of the target.

Lemma 6 *If, regardless of the location of the target in R_{n-1}, placing q_n results in an accuracy gain r_{n-1}/r_n that is at least $1+\frac{d}{2}$, then at most one new d-close pair is created.*

Lemma 7 *If, regardless of the location of the target in R_{n-1}, placing q_n results in an accuracy gain of at least $2+\frac{d}{2}$, then no new d-close pairs are created.*

Lemma 8 *For any h and d such that $h \geq 2+\frac{d}{2}$ and $\frac{2}{h} \leq d \leq 2$ and for any $c > 1$, the following holds: if, regardless of the location of the target in R_{n-1}, placing q_n results in an accuracy gain r_{n-1}/r_n that exceeds h, then, for certain locations of the target, there is a positive integer k such that the accuracy gain r_{n-1}/r_n is less than $c^k/\left((c-1)\cdot(1-\frac{2}{h})\right)$ and the number of d-close pairs decreases by at least k.*

We can now derive the following formula that expresses the cumulative effects of Lemmas 6, 7 and 8:

Lemma 9 *Let a_n and $p_n(d)$ be the accuracy and the number of d-close pairs obtained after placing n query points. For any h, d, c, and w such that $h \geq 2+\frac{d}{2}$ and $\frac{2}{h} \leq d \leq 2$ and $1 < c \leq w$, we have, for certain locations of the target and for each n:*

$$a_n w^{p_n(d)} \leq b^n,$$

where b is the maximum of $\left(1+\frac{d}{2}\right)w^2$, $\left(2+\frac{d}{2}\right)w$, h, and $c/\left(w(c-1)(1-\frac{2}{h})\right)$.

Lemma 9 is proven by induction. For each query point, we distinguish four cases depending on which of the previous three lemmas is the last one that applies. In each case we find that, for some locations of the target the left-hand side of the above inequality increases at most as much as the right-hand side.

To get the most out of Lemma 9, we choose h as the largest root of h^3-4h^2+h+1, $d=\frac{2}{h}$, and $c=w=h/(2+\frac{1}{h})$, and we get $b=h\approx 3.6511$. Since $p_n(d)$ cannot be negative, this implies that there are targets for which $a_n \leq a_n w^{p_n(d)} \leq b^n$ for all n, and thus, such targets are not located with accuracy better than $O(b^n)$. This concludes the proof of Theorem 5.

4 Conclusions and outlook

For the one-shot variant of our location problem, we presented an upper bound and a constructive lower bound on the accuracy that can be achieved. The remaining constant-factor gap between these bounds may be due to the fact that most bisectors formed by pairs of both tightly or both widely spaced points do not contribute anything to the accuracy. However, the upper bound might also be improved.

For the incremental variant, we obtained non-trivial upper and lower bounds but a gap that is exponential in n still remains. Our search strategy does not seem to leave any space for substantial improvements, so further progress must come from an entirely new search strategy, or from tightening the upper bound. Observe that in each step, our search strategy only uses the bisectors that are formed with the two previously placed query points that are closest to the target. If we could prove that there is an optimal search strategy with this property, then this would immediately improve the upper bound to $O(3^n)$.

To search a d-dimensional unit cube, we could place roughly n/d query points on each coordinate axis in a round-robin fashion; on each axis, we place the points according to the one-shot, incremental, or on-line incremental strategy. Thus, where we obtain accuracy $h(n)$ in one dimension, we can pinpoint a target to a cube of width $1/h(\lfloor n/d \rfloor)$ in d dimensions. This approach, however, fails to take advantage of bisectors between query points placed on different axes, not to mention query points placed more freely. Indeed, for $d=2$, there are better solutions for $n \in \{3,4\}$.

Acknowledgements

We thank all participants of the 2016 Lorentz Center Workshop on Search Games, and, in particular, Bengt Nilsson and Endre Csóka, for their invaluable contributions in defining the problem and initial solutions.

References

[1] H.-K. Ahn, S.-W. Cheng, O. Cheong, M. Golin, and R. van Oostrum. Competitive facility location: the Voronoi game. *Th. Comp. Sc.*, 310(1):457–467, 2004.

[2] R. Cole R. and Ch. K. Yap. Shape from probing. *J. of Algorithms*, 8(1):19– 38,1987.

[3] J. Kahn and M. Saks Balancing poset extensions *Order*, 1: 113– 126, 1984.

[4] J. Matoušek. *Lectures on discrete geometry*, Springer, New York, 2002.

[5] S. Skiena. Interactive reconstruction via geometric probing. *Proc. of the IEEE*, 80(9):1364–1383, 1992.

Distance Measures for Embedded Graphs

Maike Buchin* Stef Sijben* Carola Wenk†

Abstract

We introduce new distance measures for comparing plane embedded graphs based on strong and weak Fréchet distance. These distances use both the structure and the embedding of the graphs. We present an algorithm to compute the weak graph distance of two straight-line graphs of complexity n each in $O(n^2 \log n)$ time and a randomized algorithm that computes the graph distance in $O(n^{5/2+\delta})$ expected time for any $\delta > 0$.

1 Introduction

Embedded graphs occur often in applications, e.g., as road networks. Often they need to be compared. A few different approaches have been proposed in the literature for comparing such graphs. Subgraph-isomorphism as well as edit distance approaches [7] are NP-hard. Other distance measures compare all paths [1] or random samples of shortest paths [8]. In order to capture more topological information, Biagioni and Eriksson developed a sampling-based distance measure [5] and Ahmed et al. introduced the local persistent homology distance [2]. Alt et al. defined a distance based on mapping graph traversals [3].

Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be plane embedded graphs with vertices embedded as points in the plane that are connected by straight-line edges. We define two distance measures on such graphs, based on mappings between the graphs that have small distance in the embedding space.

We consider maps $s\colon G_1 \to G_2$ that map each vertex $v \in V_1$ to a point $s(v)$ on G_2 (not necessarily a vertex) and that map each edge $\{u, v\} \in E_1$ to a simple path in G_2 with endpoints $s(u)$ and $s(v)$. We require that edges are mapped to paths such that these have small (weak) Fréchet distance [4] in the embedding space. We call the resulting definitions the *directed (strong) graph distance* δ_d and the *directed weak graph distance* $\delta_{d,w}$:

$$\delta_d(G_1, G_2) := \inf_{s:G_1 \to G_2} \max_{e \in E_1} \delta_F(e, s(e)),$$
$$\delta_{d,w}(G_1, G_2) := \inf_{s:G_1 \to G_2} \max_{e \in E_1} \delta_{wF}(e, s(e)),$$

where s ranges over all maps as defined before and δ_F denotes the Fréchet distance and δ_{wF} the weak Fréchet distance of the edge e and its image $s(e)$ interpreted as curves in the embedding space. The Fréchet distance and the weak Fréchet distance, also known as dog-leash distances, are popular distance measures for curves. For two polygonal curves of complexity n they can be computed in $O(n^2 \log n)$ time [4]. We define the undirected distance δ_u between G_1 and G_2 as the maximum of the directed distances, i.e., $\delta_u(G_1, G_2) = \max(\delta_d(G_1, G_2), \delta_d(G_2, G_1))$ and analogously we define the weak undirected distance $\delta_{u,w}$.

We observe that the graph distance and weak graph distance are pseudo-metrics, i.e., are symmetric and fulfill the triangle inequality. The (weak) graph distance between two graphs is zero if the (weak) Fréchet distance between their embeddings is zero.

Note that we do not require the mappings to be injective or surjective, and the mappings in the two directions may be far from inverse to each other.

A similar directed distance measure for graphs was considered by Alt et. al. [3]. They define the *traversal distance* of two connected embedded straight-line graphs G_1, G_2 as

$$\delta_T(G_1, G_2) = \inf_{f,g} \max_{t \in [0,1]} ||f(t) - g(t)||.$$

where f ranges over all traversals of G_1 and g over all partial traversals of G_2. A *traversal* of G_1 is a continuous, surjective map $f\colon [0,1] \to G_1$, and a *partial traversal* of G_2 is a continuous map $g\colon [0,1] \to G_2$.

*Department of Mathematics, Ruhr-Universität Bochum, {Maike.Buchin,Stef.Sijben}@rub.de

†Department of Computer Science, Tulane University, cwenk@tulane.edu. Work by Carola Wenk was supported by National Science Foundation grant CCF-1618469.

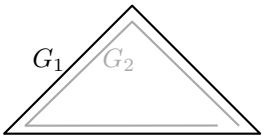


Figure 1: Two graphs with small traversal but large (weak) graph distance.

We observe that our distance measures are stronger distances in the sense that $\delta_T(G_1, G_2) \leq \delta_{d,w}(G_1, G_2) \leq \delta_d(G_1, G_2)$, if G_1 and G_2 are connected. This follows because a map in our sense maps any traversal of G_1 to a partial traversal of G_2 with distance $\leq \varepsilon$, although the traversal might need to be slightly altered for the weak distance. Figure 1 shows two graphs that have large graph distance but small traversal distance.

2 Algorithms

First we consider the decision problem, that is deciding if the (weak) directed distance between two plane graphs G_1 and G_2 is at most a given distance ε. The undirected distance is decided by deciding both directed distances. We always assume that G_1 and G_2 are plane straight-line graphs of complexity n each. Due to space restrictions we omit proofs in this extended abstract.

2.1 Valid placements

A mapping realizing a given distance maps each vertex of G_1 to a point in G_2, and it maps each edge of G_1 to a path in G_2. The algorithm iterates over all vertices and edges of G_1, maintaining all possible *placements* of vertices and of edges.

Definition 1 *An* ε-placement *of a vertex v is a connected component of G_2 restricted to the ε-ball $B_\varepsilon(v)$ around v. An* ε-placement *of an edge $e = \{u, v\}$ is a path P in G_2 with endpoints on placements C_u of u and C_v of v such that $\delta_F(e, P) \leq \varepsilon$; in that case we say that C_u and C_v are* reachable *from each other. An* ε-placement *of G_1 is a map $s\colon G_1 \to G_2$ such that s maps each edge e of G_1 to an ε-placement.*

Note that deciding whether $\delta_d(G_1, G_2) \leq \varepsilon$ is equivalent to deciding whether an ε-placement of G_1 exists. In the context of the decision algorithm, we assume ε is fixed and we use the term *placement* for an ε-placement and we use *vertex-placement* to refer to a placement of a vertex. Weak ε-placements for edges and graphs are defined analogously.

Our algorithm iterates once over all vertices and edges of G_1. First, we iterate over all vertices $v \in V_1$ and compute all their placements. Then we iterate over all edges $e = \{u, v\} \in E_1$ to determine which vertex-placements allow a placement of e, i.e., we search for placements C_u of u and C_v of v such that there is a path P in G_2 between C_u and C_v with $\delta_F(e, P) \leq \varepsilon$ (or $\delta_{wF}(e, P) \leq \varepsilon$).

This problem is algorithmically simpler for the weak Fréchet distance: Here, it suffices for P to stay within an ε-tube $T_\varepsilon(e)$ around e, which contains all points whose distance to e is $\leq \varepsilon$. For the strong graph distance, we must ensure that the Fréchet distance between e and P is at most ε, i.e., a continuous and monotone map exists from e to P that maintains a distance of at most ε. This can be checked using the original dynamic programming algorithm [4] for computing the Fréchet distance.

Finally, when all vertices and edges have been processed, the algorithm must decide whether G_1 as a whole can be mapped to G_2 and construct a mapping if one exists. For this, *invalid* placements of vertices are pruned while processing vertices and edges of G_1.

Definition 2 *An ε-placement C_v for a vertex v is* valid *if for every u adjacent to v there exists an ε-placement C_u of u such that C_v and C_u are connected by a (weak) ε-placement of the edge $\{u, v\}$. Otherwise, C_v is* invalid.

2.2 Computing Edge Placements

To decide which placements of vertices u and v incident to an edge e are valid, we compute which vertex-placements of v are reachable from those of u (and vice versa). Using this information, we can decide which vertex-placements are valid and which become invalid after removing some vertex-placements in the final step of the algorithm.

For the weak graph distance, we need to find all pairs of placements of u and placements of v that can reach one another using paths contained in $T_\varepsilon(\{u, v\})$. If we restrict G_2 to its intersection with the ε-tube, all vertex-placements in the same connected component

are mutually reachable. Thus, we can process each edge in linear time and space by computing for each such connected component a pair of lists containing the placements of u and v in that component, respectively. All reachability information can be computed in $O(n^2)$ time and space.

For the strong graph distance, existence of a path inside the ε-tube is not sufficient. Instead, every placement of u stores a list of all placements of v that are reachable. The reachability information can be computed by running a graph exploration starting from each placement, which terminates if the search leaves the ε-tube or violates the Fréchet distance. This method runs a search for every placement of the start vertex and thus needs $O(n^2)$ time per edge of G_1. Since the connectivity is explicitly stored as pairs of placements that are mutually reachable, it needs $O(n^2)$ space per edge. Hence in total over all edges $O(n^3)$ time and space are needed.

2.3 Solving the Decision Problem

To compute a placement of G_1 we can use the reachability information about vertex-placements to prune invalid placements. A placement C_v of v is invalid if there exists a neighbour u of v such that C_v can reach no placements of u. All invalid placements are deleted, which might cause other placements to become invalid, as they were only connected to placements that no longer exist. Thus, the algorithm needs to iteratively delete invalid placements and then check which placements become invalid as a result. This is repeated until no invalid placements exist. After this step all remaining placements are valid.

Obviously $\delta_d(G_1, G_2) > \varepsilon$ if there is a vertex that has no valid placement. We show that the existence of at least one valid placement for each vertex is a sufficient condition for $\delta_d(G_1, G_2) \leq \varepsilon$.

Lemma 1 *If every vertex of G_1 has at least one valid ε-placement, then G_1 has a valid ε-placement. Thus $\delta_d(G_1, G_2) \leq \varepsilon$.*

The idea of the proof is that superfluous valid placements of a vertex can be removed without causing any vertex to lose all its valid placements. If every vertex has exactly one valid placement, it is easy to show that a placement of G_1 exists.

Initially there are $O(n^2)$ vertex-placements, each of which may be deleted once.

In the weak version, reachability is stored using connected components inside the ε-tube $T_\varepsilon(\{u, v\})$. If a placement C_v is deleted, it is deleted from the list of its component in $T_\varepsilon(\{u, v\})$. If the component no longer contains any placements of v, then all placements of u in that component become invalid. A placement C_v is deleted at most once, and on deletion it must be removed from one list for every edge incident to v. Thus, the time for pruning C_v is $O(\deg(v))$. Since G_1 is planar, the average degree is constant. Thus, all invalid placements can be pruned in $O(n^2)$ time.

In the strong version, every placement has a list of placements to which it is connected. On deleting C_v, it must be removed from the lists of all placements C_u to which C_v is connected. Since each vertex can have $\Theta(n)$ placements and C_v may be connected to all of them, in the worst case $O(n)$ elements have to be removed for each neighbour of v. Thus, pruning a placement takes $O(\deg(v) \cdot n)$ time and pruning all invalid placements runs in $O(n^3)$ time.

Theorem 2 *Given plane graphs G_1, G_2 with up to n vertices each and $\varepsilon > 0$, the algorithms described here decide whether $\delta_{d,w}(G_1, G_2) \leq \varepsilon$ in $O(n^2)$ time and space and whether $\delta_d(G_1, G_2) \leq \varepsilon$ in $O(n^3)$ time and space.*

2.4 Faster Algorithm for Strong Distance

We now present a more efficient randomized algorithm to decide whether $\delta_d(G_1, G_2) \leq \varepsilon$.

In the initial stage, the algorithm computes a random subset of the reachable placements for all vertex-placements. These subsets are then used during the pruning stage to check whether a placement is still valid after deleting a placement of an adjacent vertex.

Fix a number $i \in \{1, \ldots, n\}$. When computing reachability information of an edge $e = \{u, v\}$, select a uniformly random set $N(C_u)$ of i reachable placements of v for each placement C_u of u. If at most i placements are reachable, $N(C_u)$ contains all of them and we say $N(C_u)$ is *complete*. C_u stores a list, containing a pointer to each placement in $N(C_u)$, as well as a pointer to each placement that selected C_u, i.e., all C_v such that $C_u \in N(C_v)$. The average number of pointers stored with C_u for the edge e is thus at most $2i$.

When deleting a placement C_u, all pointers to it are deleted as well. Since the pointers are symmetric, this can easily be done. Thus, the expected number of pointers that are removed is $\leq 2i$ for each edge incident to u. Since the average degree is constant, pruning a placement takes amortized $O(i)$ time.

If the last pointer is deleted for a placement C_u and edge e, its validity needs to be checked. If $N(C_u)$ is complete, all reachable placements were stored initially and were subsequently deleted. Thus, C_u becomes invalid and it is marked for deletion. However, if more than i reachable placements existed, only a random subset was stored and there may be more reachable placements. In this case, $G_2 \cap T_\varepsilon(e)$ is explored again to see if reachable placements of v still exist and a new list of pointers is constructed. In this re-exploration, parts of G_2 that belong to a deleted placement are treated as not belonging to any placement.

The proper choice of i ensures that new lists do not have to be constructed too often (in expectation). Choosing $i := n^{(1+\delta)/2}$ for any $\delta > 0$ leads to the following result.

Theorem 3 *Given graphs G_1, G_2 with up to n vertices each and $\varepsilon > 0$, the algorithm described in this section decides whether $\delta_d(G_1, G_2) \leq \varepsilon$ in $O(n^{5/2+\delta})$ expected time and deterministic space, for any $\delta > 0$.*

2.5 Computing the Optimal Distance

For the computation of the distance we search over a set of critical values, employing the decision algorithm in each step. The following types of critical values can occur.

- A new vertex-placement emerges: An edge is at distance ε from a vertex.
- Two vertex-placements merge: The vertex where they connect is at distance ε from a vertex.
- The (weak) Fréchet distance of a path and an edge is ε: as described in [4].

There are $O(n^2)$ many critical values of the first two types, and $O(n^3)$ many of the last. Parametric search can be used to find the distance as described in [4], leading to a running time of $O(n^2 \log n)$ for the weak graph distance and $O(n^{5/2+\delta})$ for the graph distance.

3 Conclusion

We described two new distances for comparing embedded graphs and presented efficient algorithms for computing these distances. An open question is whether the algorithm from Section 2.4 can run faster. We suspect that choosing $i = \text{polylog}(n)$ leads to an expected running time of $O(n^2 \,\text{polylog}(n))$.

Another question is whether the (weak) graph distance can be computed more efficiently for restricted graph classes like trees or paths, but we think that some of the lower bounds for Fréchet distance [6] apply here as well, so the distances cannot be computed in strongly subquadratic time.

References

[1] Mahmuda Ahmed, Brittany T. Fasy, Kyle S. Hickmann, and Carola Wenk. Path-based distance for street map comparison. *ACM Transactions on Spatial Algorithms and Systems*, 28 pages, 2015.

[2] Mahmuda Ahmed, Brittany Terese Fasy, and Carola Wenk. Local persistent homology based distance between maps. In *22nd ACM SIGSPATIAL GIS*, pages 43–52, 2014.

[3] Helmut Alt, Alon Efrat, Günter Rote, and Carola Wenk. Matching planar maps. *Journal of Algorithms*, 49(2):262 – 283, 2003.

[4] Helmut Alt and Michael Godau. Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry & Applications*, 5(1&2):75–91, 1995.

[5] James Biagioni and Jakob Eriksson. Inferring road maps from global positioning system traces: Survey and comparative evaluation. *Transportation Research Record: Journal of the Transportation Research Board*, 2291:61–71, 2012.

[6] Karl Bringmann. Why walking the dog takes time: Fréchet distance has no strongly subquadratic algorithms unless SETH fails. In *IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 661–670, 2014.

[7] Otfried Cheong, Joachim Gudmundsson, Hyo-Sil Kim, Daria Schymura, and Fabian Stehn. Measuring the similarity of geometric graphs. In *International Symposium on Experimental Algorithms*, pages 101–112, 2009.

[8] Sophia Karagiorgou and Dieter Pfoser. On vehicle tracking data-based road network generation. In *20th ACM SIGSPATIAL GIS*, pages 89–98, 2012.

Fréchet Isotopies to Monotone Curves*

Kevin Buchin† Erin Chambers‡ Tim Ophelders† Bettina Speckmann†

1 Introduction

We study the *isotopic Fréchet distance*, which is a distance measure between two curves f and g that captures one notion of an optimal morph between these two curves. The classic Fréchet distance between f and g, also called the "dog leash distance", measures the length of the shortest possible straight leash needed to connect a man and a dog which are walking forward along f and g. Any two feasible walks using such a shortest leash induce a *Fréchet matching* between f and g. One can now imagine to build a morph between f and g by sliding each point of f along the leash that connects it to its matched point on g. Such an approach will work well in unrestricted Euclidean space, however, it is not suitable for more general spaces that might contain obstacles. In the presence of obstacles the leashes of the classic Fréchet distance can jump discontinuously and hence the resulting morph would be discontinuous as well.

The *homotopic Fréchet distance* [3, 6] forces leashes to move continuously. More formally, for two curves f and $g\colon [0,1] \to \mathbb{R}^2$ in the plane a homotopy $h\colon [0,1]^2 \to \mathbb{R}^2$ is a continuous map between f and g. Such a homotopy essentially morphs one curve into the other: each point of f traces a path $h(p,\cdot)$ to a point on g. The length of a homotopy is the length of the longest such path, and a *Fréchet homotopy* is one that minimizes this length. The homotopic Fréchet distance between f and g is then the length of a Fréchet homotopy between f and g. The homotopic Fréchet distance and the classic Fréchet distance are equivalent in $\mathbb{R}^2$. The morph that results from a Fréchet homotopy is continuous, but it may change the structure of the input curves during the morph: intermediate curves can self-intersect or collapse to a point, even if f and g are simple curves.

A homotopy is an isotopy if all its intermediate curves $h(\cdot,t)$ are simple. The *isotopic Fréchet distance* measures the length of an optimal isotopy between f and g; we call an optimal isotopy a *Fréchet isotopy*. The study of Fréchet isotopies was initiated in [4]. The authors gave some simple observations and examples and showed that the isotopic Fréchet distance in the plane can be arbitrarily larger than the homotopic Fréchet distance.

Results. In this paper we revisit the isotopic Fréchet distance and refute a conjecture posed in [4]. We also give the first algorithms to compute short isotopies in some restricted cases. Specifically, we compute optimal isotopies if there is a direction in which both input curves are monotone. Furthermore, given a curve in $\mathbb{R} \times [0,\varepsilon]$ (for infinitesimally small ε), we construct an isotopy to a monotone curve using minimal length.

Related work. Closely related are morphs based on *geodesic width* [5]: the intermediate curves are not allowed to cross the input curves f and g, and they are restricted to the area between the leashes connecting the endpoints of f and g. This restriction naturally enforces intermediate curves without self-intersections since "geodesic leashes" do not cross each other. Morphs based on geodesic width minimize the maximum leash length. However, they are restricted to input curves that do not intersect each other; in contrast, Fréchet isotopies are also well-defined for input curves that intersect each other.

A variety of morphs have been considered in the graph drawing and computational geometry literature. For instance, it is well known that any two drawings of the same planar graphs can be morphed into one another. More recent work focused on bounding the number of steps in the optimal morph between any two input graphs [1, 2]. Here the intermediate curves are homeomorphic to the input and vary continuously. However, in contrast to Fréchet isotopies, the morphs do not minimize length.

2 Preliminaries

A *curve* in the plane is a continuous map $f\colon [0,1] \to \mathbb{R}^2$. We denote the x and y-coordinates of $f(p)$ by $f_x(p)$ and $f_y(p)$, respectively. A continuous nondecreasing surjection $\alpha\colon [0,1] \to [0,1]$ is called a *reparameterization* of a curve. A *homotopy* is a continuous map $h\colon [0,1] \times [0,1] \to \mathbb{R}^2$. We denote its level curves by $h_t\colon p \mapsto h(p,t)$, and say h goes from curve f to g if $h_0 = f$ and $h_1 = g$. A homotopy is an *isotopy* if each curve h_t is simple.

*K. Buchin, T. Ophelders and B. Speckmann are supported by the Netherlands Organisation for Scientific Research (NWO) under project no. 612.001.207 (K. Buchin) and project no. 639.023.208 (T. Ophelders and B. Speckmann). E. Chambers is supported in part by NSF grants IIS-1319944, CCF-1054779, and CCF-1614562.

†Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands, [k.a.buchin|t.a.e.ophelders|b.speckmann]@tue.nl

‡Department of Computer Science, Saint Louis University, Saint Louis, MO, USA, echambe5@slu.edu

A homotopy from f to g traces paths $\lambda_{h,p}\colon t \mapsto h(p,t)$ between the points $f(p)$ and $g(p)$, and such a path is traditionally referred to as a *leash*. Let the *length* of a homotopy h be the length $length(h) = \sup_p length(\lambda_{h,p})$ of its longest leash. We are interested in homotopies h minimizing this length and define the *homotopic Fréchet distance* between f and g as

$$d_{hom}(f,g) = \inf_{\substack{\alpha,\beta,h \\ h_0=f\circ\alpha,\ h_1=g\circ\beta}} length(h),$$

where h ranges over homotopies and α and β range over reparameterizations. The *isotopic Fréchet distance* d_{iso} is defined similarly, except that h ranges over isotopies.

The *Fréchet distance* $d_F(f,g) = \inf_{\alpha,\beta}\sup_p \|f\circ\alpha(p) - g\circ\beta(p)\|$ is a related measure that does not require leashes to trace out a homotopy, so each leash can be assumed to be a shortest path. The pair (α,β) is called a *matching*. We define the *cost* of a matching (α,β) between f and g as $cost_{f,g}(\alpha,\beta) = \sup_p \|f\circ\alpha(p) - g\circ\beta(p)\|$. A *Fréchet matching* between curves f and g in the plane is one with cost $d_F(f,g)$.

In the plane, the map $Aff^{f,g}(p,t) = (1-t)\cdot f(p) + t\cdot g(p)$ using line segments (shortest paths) as leashes is a homotopy since it is an affine interpolation between continuous maps. We call $Aff^{f,g}$ the *affine homotopy* from f to g, and its length is $length(Aff^{f,g}) = \sup_p \|f(p) - g(p)\|$. It follows that the homotopic Fréchet distance and the Fréchet distance are equivalent in $\mathbb{R}^2$. On the other hand, the isotopic Fréchet distance in the plane can be arbitrarily larger than the homotopic Fréchet distance [4].

We call a homotopy h from $f\circ\alpha$ to $g\circ\beta$ a *Fréchet homotopy* if $length(h) = d_{hom}(f,g)$, and call h a *Fréchet isotopy* if h is an isotopy with $length(h) = d_{iso}(f,g)$. Since every isotopy is a homotopy, we have $d_{hom}(f,g) \le d_{iso}(f,g)$ and any isotopy that is a Fréchet homotopy is also a Fréchet isotopy. However, Fréchet isotopies need not be Fréchet homotopies since there might exist a homotopy shorter than any isotopy.

For a curve f and a unit vector $(x,y) \in S^1$, we define the *directional length* of f in the direction (x,y) to be the total length that f moves *forward* in the direction of the vector, given by $length_{(x,y)}(f) = \int_0^1 \max(0, \langle \frac{\mathrm{d}f(p)}{\mathrm{d}p}, (x,y)\rangle)\mathrm{d}p$, where $\langle\cdot,\cdot\rangle$ is the inner product. We define the horizontal length of a curve as $length_{\mathrm{hor}}(f) = length_{(-1,0)}(f) + length_{(1,0)}(f)$ and define the *horizontal* homotopic and isotopic Fréchet distances using the horizontal length function. As usual, a horizontal Fréchet homotopy (respectively isotopy) is one minimizing the horizontal homotopic (respectively isotopic) Fréchet distance.

Throughout the paper, we assume all input curves to be simple.

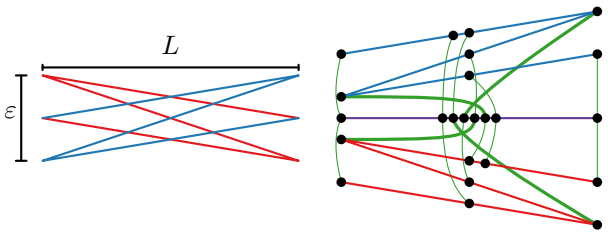


Figure 1: An isotopy of length $L/2$ (as ε approaches 0) between two 'opposite' zig-zags. The fat arcs have horizontal length roughly $L/2$, whereas the others have negligible horizontal length.

3 Disproving a conjecture

In Figure 1 we show an example of two zig-zag curves, originally presented in [4]. The Fréchet distance between these curves is at most ε, as there is a matching whose leashes are all vertical. However, this Fréchet mapping yields a homotopy that collapses the zig-zag to a flat line before re-expanding to the other zig-zag, which does not result in an isotopy, as the three segments coincide halfway along the isotopy.

In [4], the authors conjectured that the isotopic Fréchet distance between the zig-zags is $\sqrt{L^2+\varepsilon^2}$. However, the isotopy demonstrated by the green leashes on the right side of Figure 1 has length arbitrarily close to $\sqrt{L^2+\varepsilon^2}/2 + \varepsilon/2$.

We will show that the isotopy of Figure 1 is arbitrarily close to optimal. Consider a convex region D and an isotopy h between curves f and g in the plane, where the endpoints of all intermediate curves lie in D; that is, $\mathrm{Im}(\lambda_{h,0}) \subseteq D$ and $\mathrm{Im}(\lambda_{h,1}) \subseteq D$. Fix some $p \in (0,1)$ and denote by $poly_t$ the polyline with an edge from $h(0,t)$ to $h(p,t)$, and an edge from $h(p,t)$ to $h(1,t)$. Let θ_t be the (counterclockwise) angle at $h_t(p)$ between the two edges of $poly_t$ (plus a multiple of 360 degrees), such that θ_t varies continuously with t. We show in Lemma 1 that (in any isotopy from $poly_0$ to $poly_1$) the leash $\lambda_{h,p}$ must intersect D if θ_0 and θ_1 differ by at least 180 degrees, see Figure 2.

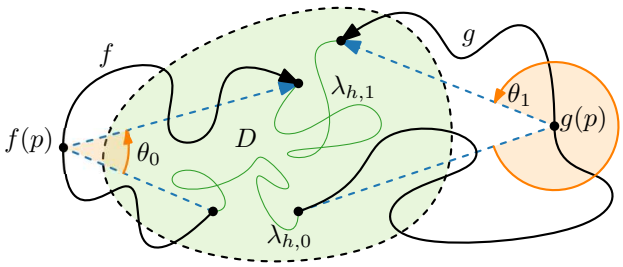


Figure 2: Curves $f = h_0$, $g = h_1$ and polylines $poly_0$ and $poly_1$ with endpoints in convex region D.

Lemma 1 *If f is isotopic to $poly_0$ relative to its vertices*[1] *and g is isotopic to $poly_1$ relative to its vertices, and $|\theta_1 - \theta_0| \geq 180$, then $h(p,t) \in D$ for some t.*

Proof. Because f and g are isotopic to $poly_0$ and $poly_1$ respectively (relative to their vertices), we may assume without loss of generality that $0 \leq \theta_0 < 360$ and $0 \leq \theta_1 < 360$. Because θ_t varies continuously, we have by the intermediate value theorem that $\theta_t = 180$ for some $t \in [0,1]$. Hence, $h(p,t)$ lies on the line segment between $h(0,t) \in D$ and $h(1,t) \in D$. By convexity, this segment lies completely in D, so $h(p,t) \in D$. □

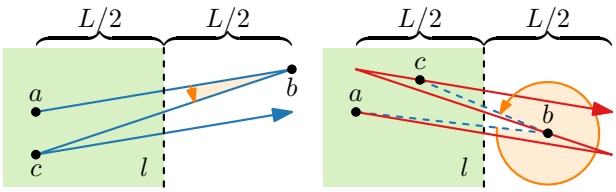


Figure 3: The vertices and region D used to obtain a lower bound for the curves of Figure 1.

Using Lemma 1, we can show that our isotopy for the zig-zags of Figure 1 is optimal as ε approaches 0. For this, we show that any Fréchet isotopy has length at least $L/2$. Assume that the zig-zags f and g are parameterized such that an isotopy h of length less than $L/2$ between them exists. Let l be the vertical line centered between the vertices, such that each vertex has distance $L/2$ to l, and let D be the half-plane to the left of l, see Figure 3. Let $f(a)$, $f(b)$ and $f(c)$ be the first three vertices of f. If $length(h) < L/2$, the leashes $\lambda_{h,a}$ and $\lambda_{h,c}$ lie completely inside D, and $\lambda_{h,b}$ lies completely outside D. Since $a < b < c$ and $g(b) \notin D$, a and c lie in different components of $g^{-1}(D)$. Isotopy h induces a restricted isotopy between the subcurves of f and g from a to c, and these subcurves satisfy the conditions required by Lemma 1. Therefore, $\mathrm{Im}(\lambda_{h,b})$ intersects D, so $length(h) \geq L/2$.

4 Isotopies between monotone curves

A curve f is *strictly x-monotone* if $f_x(p) < f_x(p')$ for all $p < p'$. Lemma 2 implies that for such curves, the isotopic and homotopic Fréchet distances are equal.

Lemma 2 *For strictly x-monotone curves f and g, each curve h_t of $h = Aff^{f,g}$ is strictly x-monotone.*

Proof. Recall that $h_t(p) = (1-t) \cdot f(p) + t \cdot g(p)$. Consider the x-coordinates $x_t(p)$ of $h_t(p)$ and $x_t(p')$ of $h_t(p')$ for $p < p'$. Let $s_t = x_t(p') - x_t(p)$. Because f and g are strictly x-monotone, we have $x_0(p) < x_0(p')$ and $x_1(p) < x_1(p')$, so $s_0 > 0$ and $s_1 > 0$. Since s is affine, we have $s(t) > 0$ for $t \in [0,1]$, so $x_t(p) < x_t(p')$. Hence, each level curve h_t is strictly x-monotone. □

[1] That is, there exists an isotopy from f to $poly_0$ that does not move $f(0)$, $f(p)$ or $f(1)$.

Theorem 3 *For strictly x-monotone curves, the homotopic and isotopic Fréchet distances are equivalent.*

5 Isotopies to monotone curves

In this section, we consider the problem of monotonizing curves using minimal horizontal movement. Specifically, we show how to construct a short isotopy from an input curve to some x-monotone curve. In Section 5.1 we argue that this problem is interesting already if we measure only horizontal length by showing that an optimal isotopy may have non-monotone leashes, even though we can choose any x-monotone target curve. We construct an isotopy of minimal horizontal length to an x-monotone curve in Section 5.2. We first give a lower bound for homotopies to x-monotone curves.

Lemma 4 *For any homotopy h from f to any x-monotone curve g, $length_{\mathrm{hor}}(h) \geq \frac{1}{2} \sup_{p \leq p'} f_x(p) - f_x(p')$.*

Proof. We have $length_{\mathrm{hor}}(\lambda_{h,p}) \geq |f_x(p) - g_x(p)|$ and because g is x-monotone, $g_x(p) \leq g_x(p')$. Because $length_{\mathrm{hor}}(h) \geq length_{\mathrm{hor}}(\lambda_{h,p}) \geq f_x(p) - g_x(p)$ and $length_{\mathrm{hor}}(h) \geq length_{\mathrm{hor}}(\lambda_{h,p'}) \geq g_x(p') - f_x(p')$, we have $2 \cdot length_{\mathrm{hor}}(h) \geq f_x(p) - g_x(p) + g_x(p') - f_x(p') \geq f_x(p) - f_x(p')$. □

5.1 Non-monotone isotopies

Consider the curve $f = h_0$ of Figure 4 with $f(0) = p_0$ and $f(1) = p_5$, morphing into an x-monotone curve $g = h_1$ as depicted. No matter how small we pick $\varepsilon > 0$, the depicted isotopy has length at most $r + \varepsilon$ if $w < r$. By Lemma 4, there exists no homotopy to an x-monotone curve of length less than r, even if we pick a different x-monotone curve g. In this context, the lemma states that because $f_x(p_1) \geq f_x(p_2) + 2r$ and $g_x(p_1) \leq g_x(p_2)$, one of the leashes λ_{h,p_1} or λ_{h,p_2} has length at least r in any homotopy h.

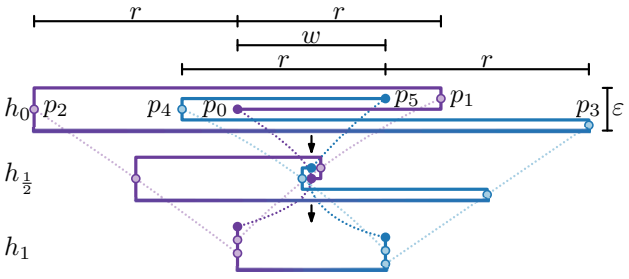


Figure 4: A curve for which any Fréchet isotopy to an x-monotone curve moves some point for distance $w/2 - \varepsilon$ in both the positive and the negative x-direction.

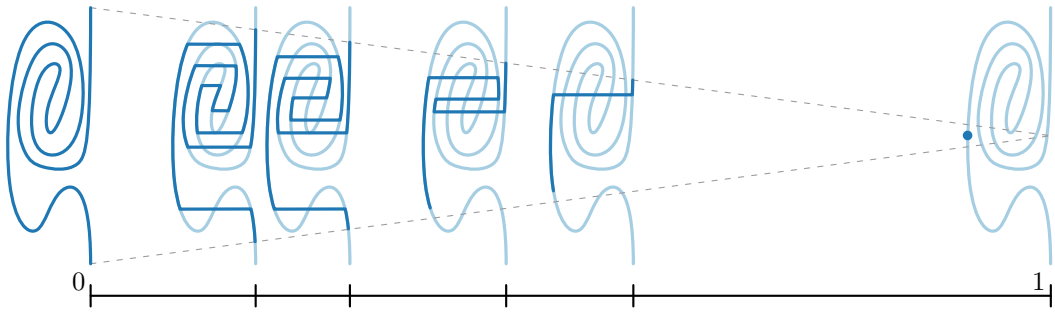


Figure 5: The major critical events of our isotopy (rotated 90 degrees) on a spiral.

What makes this instance interesting is that any optimal isotopy moves some points in both the forward (positive) and backward (negative x-direction) for a considerable distance. Formally, for any isotopy h from f to g, we have both $length_{(1,0)}(\lambda_{h,p}) \geq w/2 - \varepsilon$ and $length_{(-1,0)}(\lambda_{h,p}) \geq w/2 - \varepsilon$ for some p. In particular, consider the two endpoints $f(p_0)$ and $f(p_5)$ in Figure 4. Based on Lemma 1, these points must 'untangle' with respect to each other somewhere in the isotopy h. For this, the x-coordinates of $h_t(p_0)$ and $h_t(p_5)$ must be equal for some value of t, say for $t = t^*$. Because g is x-monotone, and an optimal isotopy has length at most $r + \varepsilon$, we have $g_x(p_0) \leq g_x(p_2) \leq f_x(p_2) + r + \varepsilon = f_x(p_0) + \varepsilon$ and symmetrically $f_x(p_5) - \varepsilon = f_x(p_3) - r - \varepsilon \leq g_x(p_3) \leq g_x(p_5)$.

Let $\gamma = h_{t^*}$ and $x^* = \gamma_x(p_0) = \gamma_x(p_5)$. Since $f_x(p_5) - f_x(p_0) = w$, we have $x^* - f_x(p_0) \geq w/2$ or $f_x(p_5) - x^* \geq w/2$. If $x^* - f_x(p_0) \geq w/2$, we also have $g_x(p_0) - x^* \geq w/2 - \varepsilon$, so p_0 moves forward for a distance of at least $w/2$ and backwards for a distance of at least $w/2 - \varepsilon$. Otherwise, $f_x(p_5) - x^* \geq w/2$, and p_5 moves backwards for $w/2$ and forwards for $w/2 - \varepsilon$. The total distance such points move approaches $d_{iso}(f, g)$ as ε approaches 0.

5.2 Shrinking based isotopies

For a curve f, we define an isotopy $Shr^f(p, t) = \mathcal{P}_{|\mathrm{Im}(f_x)|t/2}(f)(p)$, where $\mathcal{P}_l(f)$ is a curve intuitively obtained from f by moving all local maxima of f_x in the negative x-direction, and all local minima of f_x in the positive x-direction for distance l. Define $N_l^f(p)$ to be the component of $f^{-1}((f_x(p) - l, f_x(p) + l))$ containing p; that is, the subpath of f reachable from $f(p)$ using only x-coordinates at distance less than l from $f_x(p)$. Let l^* be the horizontal length of the longest monotone[2] subpath of f. We can define $\mathcal{P}_l(f)$ more formally by recursively defining $\mathcal{P}_l(f) = \mathcal{P}_{l-l^*}(\mathcal{P}_{l^*}(f))$ if $l > l^*$; and if $l \leq l^*$, replacing, for each minimum or maximum p of f_x, the arc $N_l^f(p)$ of f by a vertical segment between its endpoints with f_x at distance l from $f_x(p)$ (or by the endpoint if only one such endpoint exists).

[2]Monotone in either the positive or the negative x-direction.

In the full paper, we show that after infinitesimal perturbation, Shr^f is an isotopy. Figure 5 illustrates its behavior for a curve based on an example from [4].

5.3 Optimality of the isotopy to a monotone curve

Lemma 4 gave us a lower bound on $length_{\mathrm{hor}}(h)$ for a homotopy h turning f into an x-monotone curve. Theorem 5 (proof in the full paper) tells us that the horizontal length of the isotopy we construct matches this lower bound, meaning that the isotopy yields an x-monotone curve using minimal horizontal movement.

Theorem 5 *$Shr^f(t)$ yields an x-monotone curve at $t = |\mathrm{Im}(f_x)|^{-1}\frac{1}{2}\sup_{p \leq p'} f_x(p) - f_x(p')$, using only $\frac{1}{2}\sup_{p \leq p'} f_x(p) - f_x(p')$ horizontal movement.*

References

[1] P. Angelini, G. Da Lozzo, G. Di Battista, F. Frati, M. Patrignani, and V. Roselli. Morphing planar graph drawings optimally. In *Automata, Languages, and Programming*, *LNCS* 8572, pages 126–137, 2014.

[2] P. Angelini, F. Frati, M. Patrignani, and V. Roselli. Morphing planar graph drawings efficiently. In Proc. *21st International Symposium on Graph Drawing*, *LNCS* 8242, pages 49–60, 2013.

[3] E. W. Chambers, É. C. de Verdière, J. Erickson, S. Lazard, F. Lazarus, and S. Thite. Homotopic Fréchet distance between curves or, walking your dog in the woods in polynomial time. *Computational Geometry*, 43(3), 2010.

[4] E. W. Chambers, D. Letscher, T. Ju, and L. Liu. Isotopic Fréchet distance. In *Canadian Conference on Computational Geometry*, 2011.

[5] A. Efrat, L. J. Guibas, S. Har-Peled, J. S. B. Mitchell, and T. M. Murali. New similarity measures between polylines with applications to morphing and polygon sweeping. *Discrete & Computational Geometry*, 28(4):535–569, 2002.

[6] S. Har-Peled, A. Nayyeri, M. Salavatipour, and A. Sidiropoulos. How to walk your dog in the mountains with no magic leash. In *Proc. 28th Symposium on Computational Geometry*, pages 121–130, 2012.

Computing representative networks for braided rivers*

M. Kleinhans† M. van Kreveld‡ T. Ophelders§ W. Sonke§ B. Speckmann§ K. Verbeek§

1 Introduction

Geomorphology is the study of the shape of natural terrains and the processes that create them. One of these processes is erosion due to water flow. The combination of terrain shape and water flow gives rise to various computational problems that have been studied in geomorphology, geographic information science (GIS), and computational geometry.

One prominent problem is the computation of drainage networks (flow) [1]. Here computations are based on elevation data only and the shape of the terrain is used to determine where rivers will form (see [11] for an overview). A second problem concerns local minima. Due to erosion local minima are more rare in natural terrains than local maxima; minor local minima are often measurement errors. Such errors are clearly undesirable when studying flow on terrains and hence, minor local minima are removed by computational means [6]. A third commonly studied problem deals with watersheds and their boundaries [2, 11].

Braided rivers. The usual models for water flow in terrains allow rivers to merge, which is natural because side valleys join main valleys. And clearly, if water always follows the direction of steepest descent, a river cannot split (except due to degeneracies). Yet splitting happens in deltas and various river types, in particular braided river systems [4]. Such systems have islands called bars, separating different channels of the same river over their length after which the channels confluence. Modeling braided rivers, where channels can both split and merge, is considerably more complex than modeling standard drainage networks, where all rivers flow only downhill and do not bifurcate. In this paper we initiate the study of braided rivers from the perspective of computational geometry and topology.

To model a braided river we first need a representation of the basic geometry, independent of water level. We hence use the elevation of the river bed as a starting point. In meandering rivers the so-called *thalweg* is often used as a basic representation. The thalweg is defined as the deepest part of a continuous channel, which is a linear feature. We are striving for a similar representation for braided rivers, consisting of linear features along *lowest paths* in each channel. These linear features can merge and bifurcate, that is, they form a planar graph or network. The use of graphs to model and analyze braided rivers was recently pioneered in [7]. We define lowest paths intuitively as the paths that do not go higher than they need to go to.

A representative network for a braided river should not necessarily contain all possible channels. Topologically speaking a tiny local maximum in the river bed creates two channels. We can use persistence to simplify our input and avoid such situations. But still, too many channels may remain. We wish to select a set of channels which are sufficiently different from each other, which we model with a function (the *sand function*) that relates to the volume of sediment the river has to move before the two channels become one. More volume needs more time to be removed [5]. A bar of very small volume separating two channels requires insignificant time to be removed, but a large bar with a large volume may require multiple floods to be shaved off or cut through by a new channel, meaning that the two channels separated by this bar are significantly different.

Our objective now is to compute a representative network of channels that is optimal in some sense. We require that (*i*) each channel is *locally lowest*, (*ii*) any two channels are sufficiently *different* (specified by a parameter δ and the sand function), and (*iii*) the representative network is *maximum*. Unfortunately, solving this problem exactly is NP-hard. As an alternative we first compute a *striation*: a left-bank-to-right-bank sequence of non-crossing paths for the whole river bed. We then define a sand function to measure how different two channels are, which allows us to extract a representative network from the striation using a greedy algorithm. Due to space restrictions we present only one heuristic for the striation and one model for the sand function in this short abstract. The other heuristics, models, and omitted proofs can be found in the full version of the paper.

*T. Ophelders, W. Sonke and B. Speckmann are supported by the Netherlands Organisation for Scientific Research (NWO) under project no. 639.023.208, and K. Verbeek under project no. 639.021.541. M. Kleinhans is supported by the Dutch Technology Foundation STW (grant Vici 016.140.316/13710, which is part of the Netherlands Organisation for Scientific Research (NWO), and is partly funded by the Ministry of Economic Affairs)

†Faculty of Geosciences, Utrecht University, The Netherlands, m.g.kleinhans@uu.nl

‡Dep. of Information and Computing Sciences, Utrecht University, The Netherlands, m.j.vankreveld@uu.nl

§Dep. of Mathematics and Computer Science, TU Eindhoven, The Netherlands, [t.a.e.ophelders|w.m.sonke|b.speckmann|k.a.b.verbeek]@tue.nl

2 Definitions and problem statement

Let $G = (V, E)$ be a triangulation of a topological disk M in the plane, and let $h\colon M \to \mathbb{R}$ be the *height* of the points in M, where the edges and triangles of G interpolate h linearly between its vertices. So G can be viewed as a simplicial 2-complex in $\mathbb{R}^3$ by adding h as a third dimension. Let $\sigma \in V$ be a *source* and $\tau \in V$ be a *sink*, both on the boundary ∂M. We refer to (G, h) as a *terrain*, and to the volume $\{(x, y, z) \mid (x, y) \in M, z \in \mathbb{R}, z \leq h(x, y)\}$ as *sand*.

Let $\pi^-_{\partial M}$ and $\pi^+_{\partial M}$ (respectively clockwise and counterclockwise) be the two paths from σ to τ along the boundary of M. We call a path π from σ to τ *semi-simple* if it has no self-intersections, but it may coincide with itself, see the red path in Figure 1. Let $\mathcal{P}$ be a set of semi-simple, pairwise non-crossing paths from σ to τ along edges of G. For two semi-simple paths π_0 and π_1 in $\mathcal{P}$, let $D(\pi_0, \pi_1)$ be the region bounded by and including π_0 and π_1. So two semi-simple paths π_0 and π_1 from σ to τ have no proper crossings if and only if $D(\pi^+_{\partial M}, \pi_0) \subseteq D(\pi^+_{\partial M}, \pi_1)$ or $D(\pi^+_{\partial M}, \pi_1) \subseteq D(\pi^+_{\partial M}, \pi_0)$.

We measure the similarity between two paths using a *sand function* $d\colon \mathcal{P} \times \mathcal{P} \to \mathbb{R}$, and we say a path π_0 is δ-*dissimilar* to π_1 if and only if $d(\pi_0, \pi_1) \geq \delta$. Since it is generally not a metric, we do not call d a distance.

Our goal is to compute a non-crossing set of paths $\Pi \subseteq \mathcal{P}$ that represent channels in a river. For that we use lowest paths, paths that minimize the distance spent at high elevations. Formally, for a path π, let $\rho(\pi, z)$ be the length of π for which the height is at least z. We say a path π_0 is *lower* than π_1 if and only if there exists a $z^* \in \mathbb{R}$, such that for all $z \geq z^*$, $\rho(\pi_0, z) = \rho(\pi_1, z)$ and for all $\varepsilon > 0$, there is some $z' \in (z^* - \varepsilon, z)$ with $\rho(\pi_0, z') < \rho(\pi_1, z')$ [8]. We call Π a δ-*network* if no pair of paths $\pi_0, \pi_1 \in \Pi$ has proper crossings, and $d(\pi_0, \pi_1) \geq \delta$ if π_1 is lower than π_0. A delta-network is representative unless replacing a subset of paths by a lower path yields a delta-network.

We assume that all vertices in our terrain have different height, and all edges have different slope. We attach a vertex v_∞ to the boundary of the terrain that is higher than all other vertices. Given such a modified terrain, a source σ, a sink τ, a parameter δ, and a sand function d, we study the problem of computing a representative δ-network over the edges of the terrain.

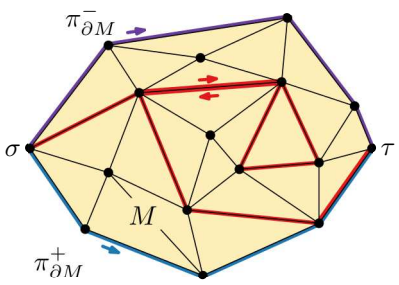


Figure 1: The disk M and three paths of $\mathcal{P}$ without proper crossings, including the two paths $\pi^-_{\partial M}$ and $\pi^+_{\partial M}$ and a backtracking path.

3 Morse-Smale complex and lowest paths

Let $\mathbb{M}$ be a smooth, compact 2-dimensional manifold without boundary, and let $h\colon \mathbb{M} \to \mathbb{R}$ be a height function on $\mathbb{M}$. A point p on $\mathbb{M}$ is *critical* with respect to h if all partial derivatives vanish at p. Otherwise, p is called *regular*. There are three types of critical points: (local) minima, (local) maxima, and saddle points. For each regular point p we can define the path of *steepest ascent* (or *steepest descent* in the opposite direction) as the path that follows the gradient of h at p. These paths are also known as *integral lines* and are open at both ends, with at each end a critical point. Using these integral lines, we can subdivide $\mathbb{M}$ as follows: two regular points p and q belong to the same cell if the integral lines through p and q end at the same critical points on both sides. The resulting complex is known as the *Morse-Smale complex*, or MS-complex in short. Note that if one of the endpoints of an integral line is a saddle point, then the corresponding cell is 1-dimensional. We refer to 2-dimensional cells of the MS-complex as *MS-cells*, and 1-dimensional cells as *MS-edges*. It can be shown [3] that every MS-cell is a quadrilateral with a minimum, a saddle, a maximum, and again a saddle along the boundary of the cell.

Note that our paths need to follow the edges of the terrain. Therefore, instead of the standard MS-complex, we use (a subset of) a quasi MS-complex as defined in [3], which does follow the edges of the terrain. Let v be a vertex, and let $S(v)$ be the *edge star* of v consisting of the set of edges incident to v. The *lower edge star* $S^{\downarrow}(v)$ consists of the subset of edges whose endpoints are lower than v with respect to h. Symmetrically, we can define the upper edge star $S^{\uparrow}(v) = S(v) \setminus S^{\downarrow}(v)$. The lower edge star can naturally be subdivided into *wedges* of consecutive edges in $S^{\downarrow}(v)$ separated by edges in $S^{\uparrow}(v)$. Given these definitions, we construct a *descending* [10] quasi MS-complex as follows. From every saddle point v we construct a steepest descent path in every wedge of $S^{\downarrow}(v)$ until it reaches a minimum. Note that steepest descent paths may partly overlap, but cannot cross. The cells of this complex are bounded by alternating minima and saddle points, and every cell contains exactly one maximum. In the remainder of this paper we refer to the descending quasi MS-complex as constructed above simply as the MS-complex, unless stated otherwise. The same rule applies to the components of the complex, namely the MS-cells and MS-edges.

The relation between lowest paths and the MS-complex is summarized in the following lemma.

Lemma 1 *The lowest path between two vertices u and v in G follows MS-edges, except for the head and tail of the path, which follow steepest descent paths.*

Due to this relation, we refer to paths on the MS-edges as *locally lowest* paths. As a result, all paths in a representative δ-network must follow MS-edges.

4 Striation

Consider an input terrain consisting of a sequence of pyramids with different heights as shown in Figure 2, where all non-peak vertices are at height 0. Let π_0 and π_1 be two paths from source to sink at height 0 and let P be the set of pyramids in $D(\pi_0, \pi_1)$. Then, for any "reasonable" sand function, $d(\pi_0, \pi_1) = \sum_{p \in P} vol(p)$. It is now easy to see that computing a representative δ-network for a terrain of this type is NP-hard by reduction from Partition.

To make the problem tractable, we put a restriction on the paths that can be used in a representative δ-network. We use a *striation*: a left-bank-to-right-bank sequence of non-crossing paths for the whole terrain. Formally, a striation $\mathcal{S}$ is an ordered set of non-crossing paths $\mathcal{S} = \{\pi_0, \ldots, \pi_r\}$ from σ to τ with $\pi_0 = \pi^-_{\partial M}$ and $\pi_r = \pi^+_{\partial M}$. Every path in a striation must be composed of MS-edges and between every two consecutive paths π_i and π_{i+1} there can be at most one MS-cell and possibly several one-dimensional features. The one-dimensional features arise from overlapping MS-edges or from the way the striation is computed. We then restrict a representative δ-network to choose paths only from the striation.

Computing a striation. The hardness result of the previous section implies that computing a striation that contains a representative δ-network (with the most lowest paths) is NP-hard. We therefore use a heuristic to compute a striation. Our heuristic uses the persistence of local maxima, which can easily be computed from the standard MS-complex [3].

The first path π is obtained by computing the lowest path from source to sink that passes through the maximum q with the highest persistence (excluding v_∞). Since π actually consists of two lowest paths (from source to q, and from q to sink), π has the form of a path π' with a special vertex v from which there is a path to q and back to v (Lemma 1). We now subdivide G as follows. Let c be the MS-cell containing q, and let

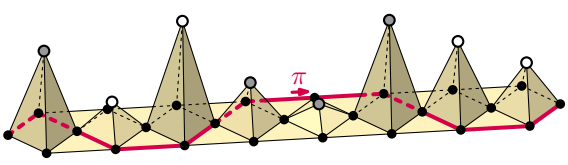


Figure 2: The path π forms a partition of the total sand volume.

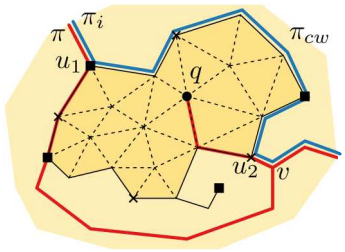


Figure 3: The lowest path π through q and π_0.

u_1 and u_2 be the first and last vertices of π that are on the boundary of c, respectively (see Fig. 3). Furthermore, let π_{cw} and π_{ccw} be the paths between u_1 and u_2 along the boundary of c in clockwise and counterclockwise direction, respectively. We can now obtain the path π_i as the concatenation of the subpath of π from σ to u_1, the path π_{cw}, and the subpath of π from u_2 to τ. Similarly, we can obtain π_{i+1} by replacing π_{cw} by π_{ccw} in π_i. If $u_1 = u_2$, then either π_i or π_{i+1} may backtrack from u_1. In that case we can replace the respective path with π'. The paths π_i and π_{i+1} subdivide G into three parts (see Fig. 4): $D_1 = D(\pi^-_{\partial M}, \pi_i)$, $D_2 = D(\pi_i, \pi_{i+1})$, and $D_3 = D(\pi_{i+1}, \pi^+_{\partial M})$. Since D_2 contains only one MS-cell, we recurse only in D_1 and D_3 to obtain $\mathcal{S}_1$ and $\mathcal{S}_3$. The final striation then consists of $\mathcal{S} = \{\pi^-_{\partial M}, \mathcal{S}_1, \pi_i, \pi_{i+1}, \mathcal{S}_3, \pi^+_{\partial M}\}$.

5 Representative network

To compute a representative δ-network conforming to a striation, we first need to define when two paths in the striation are δ-dissimilar.

Sand function. To define the dissimilarity for two paths π_i and π_{i+1} of the striation we define a sand function $d(\pi_i, \pi_{i+1})$. Intuitively, we define $d(\pi_i, \pi_{i+1})$ in such a way that it measures the volume of sand that lies between π_i and π_{i+1}. To that end we compute a homotopy (continuous morph) $\eta\colon [0,1]^2 \to M$ between π_i and π_{i+1}. We attach a height function $\zeta\colon [0,1]^2 \to \mathbb{R}$ to this homotopy, thus defining a surface in $\mathbb{R}^3$

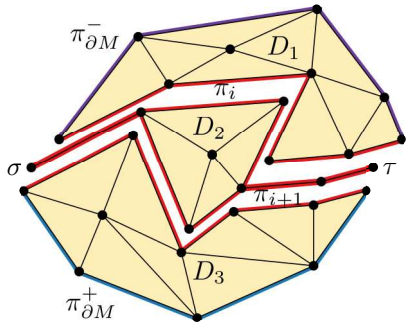


Figure 4: Splitting the triangulation by the striation paths around an MS-cell.

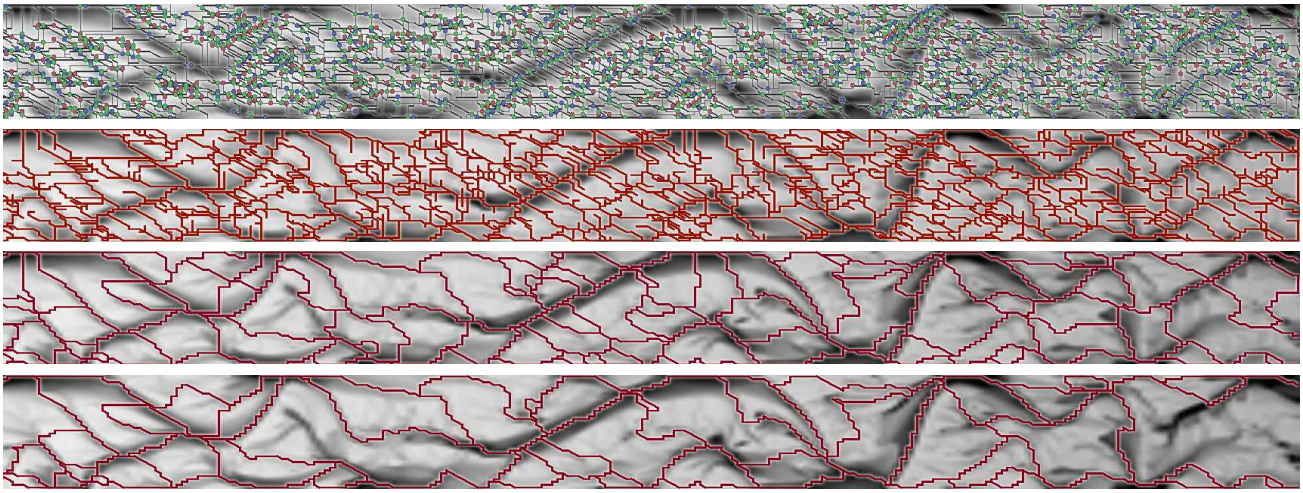


Figure 5: A modeled river with the MS-complex, the striation, and two representative δ-networks for different δ.

between π_i and π_{i+1}. We define $d(\pi_i, \pi_{i+1})$ as the volume of sand above this surface. To ensure that only sand between π_i and π_{i+1} is measured, and without multiplicity, we restrict η to be a monotone isotopy[1]. Choosing a suitable isotopy is non-trivial; details can be found in the full paper. We can extend the choices for π_i and π_{i+1} to apply to all paths π_i and π_j of the striation.

Representative network. Finally, we can easily compute a representative δ-network Π using a simple greedy algorithm. We consider all paths in the striation starting with the lowest path. We add a path π to Π if $d(\pi, \pi_i) \geq \delta$ for all $\pi_i \in \Pi$. It is easy to see that the resulting δ-network is representative.

6 Experimental results

We performed experiments on a numerically modeled river, created by a state-of-the-art model suite that is used in the civil engineering and fluvial and coastal morphology disciplines worldwide, indicating its usefulness and quality [9]. Figure 5 shows our results. First of all we observe that the representative networks capture the channels of the river very well. Second, we see that the channels which are removed in the sparser network cross big bars. Furthermore, the sparser network also avoids deep short channels which are no longer connected or did not form as a channel at all. Hence the sparser network is indeed more representative for the river than the more complex one.

References

[1] L. Arge, J. S. Chase, P. Halpin, L. Toma, J. S. Vitter, D. Urban, and R. Wickremesinghe. Efficient flow computation on massive grid terrain datasets. *GeoInformatica*, 7(4):283–313, 2003.

[2] M. de Berg and C. Tsirogiannis. Exact and approximate computations of watersheds on triangulated terrains. In *Proc. 19th ACM SIGSPATIAL Conference*, pages 74–83, 2011.

[3] H. Edelsbrunner, J. Harer, and A. Zomorodian. Hierarchical Morse complexes for piecewise linear 2-manifolds. In *Proc. 17th SoCG*, pages 70–79, 2001.

[4] A. Howard, M. Keetch, and C. Vincent. Topological and geometrical properties of braided streams. *Water Resources Research*, 6(6), 1970.

[5] M. G. Kleinhans. Flow discharge and sediment transport models for estimating a minimum timescale of hydrological activity and channel and delta formation on Mars. *J. of Geophysical Research*, 110, 2005.

[6] Y. Liu and J. Snoeyink. Flooding triangulated terrain. In *Developments in Spatial Data Handling*, pages 137–148. Springer, Berlin, 2005.

[7] W. A. Marra, M. G. Kleinhans, and E. A. Addink. Network concepts to describe channel importance and change in multichannel systems: test results for the Jamuna river, Bangladesh. *Earth Surface Processes and Landforms*, 39(6):766–778, 2014.

[8] Günter Rote. Lexicographic Fréchet matchings. In *Abstracts 30th EuroCG*, 2014.

[9] F. Schuurman, W. A. Marra, and M. G. Kleinhans. Physics-based modeling of large braided sand-bed rivers: Bar pattern formation, dynamics, and sensitivity. *J. of Geophysical Research: Earth Surface*, 118(4):2509–2527, 2013.

[10] N. Shivashankar, S. M, and V. Natarajan. Parallel computation of 2D Morse-Smale complexes. *IEEE TVCG*, 18(10):1757–1770, 2012.

[11] S. Yu, M. van Kreveld, and J. Snoeyink. Drainage queries in TINs: from local to global and back again. In *Advances in GIS Research II*, pages 829–842, 1997.

[1] The intermediate paths in the morph are semi-simple and do not cross each other.

Weighted Discrete Surveillance Tours in Simple Polygons

Bengt J. Nilsson * Eli Packer †

Abstract

The watchman route problem is a well investigated problem in which a closed tour inside a polygon that satisfies visibility constraints is optimized. We explore a new variant in which a tour needs to see a set of locations inside a polygon so that the maximum time interval in which those locations are not guarded (the delay) is minimized. We call such tours *surveillance tours*. We show that an algorithm that follows the ideas of the classic watchman problem provides a 2-approximation to our problem. We then investigate the case where each location is associated with a weight that represents the importance of not leaving the location unguarded for long periods. Thus, the tour is required to see locations with larger weights more frequently. We show that this variant is NP-hard and present two approximation algorithms.

1 Introduction

Visibility coverage of polygons with guards (mainly known as *Art Gallery* problems) have been central geometric problems for many years. Usually guards are defined as static points that see in any direction for any distance and visibility is defined by the clearance of straight lines between two features (in other words, two features see each other if the segment that connects them does not intersect (the interior of) any other feature of the input). Coverage is achieved if any point inside the polygon is visible by at least one guard. Several art gallery theorems have been proposed for different kind of settings [3].

Allowing a guard to move inside the polygons defines a related problem but yet with very different properties. Here, a set of mobile guards walk on closed cycles (also called *tours* or *routes*) so that any point inside the polygon is seen by at least one guard during its walk along the tour. The number of guards is a parameter of the problem and the measure criteria relates to the length of the tours (e.g., minimize the longest tour). Several solutions have been proposed for the case of a single mobile guard, a *shortest watchman route* in a simple polygon. The currently fastest one combines algorithms by Tan [4] and Dror *et al.* [1], to achieve asymptotic running time $O(n^4 \log n)$.

*Dept. of Computer Science, Malmö University, Malmö, Sweden. email: bengt.nilsson.TS@mah.se

†Intel Corporation, Israel. email: eli.packer@intel.com

A variant of the shortest watchman route problem is where the covering is restricted to a given finite subset of points inside the polygon instead of the entire polygon. For simple polygons, this variant is solvable in a similar way as the shortest watchman route problem; see Section 2.

We want to guard a given simple polygon $\mathbf{P}$, but rather than finding a shortest tour that covers the points of $\mathbf{P}$, we are interested in a tour that minimizes the maximum duration in which any of the points in $\mathbf{P}$ are not guarded. We call such a tour a *minimum surveillance route* for the polygon, abbreviated MSR. Kamphans and Langetepe [2] study a similar concept (*inspection paths*) but their optimization measure is the sum of the durations where features are not covered rather than the maximum duration.

We show that the two objective functions, minimize the length of the tour and minimize the maximum duration in which any point in the polygon is not guarded, have different optimal tours and that a solution to the shortest watchman route problem is a 2-approximation to the minimum surveillance route problem. We also consider a discrete version of the minimum surveillance tour problem where a given finite subset $\mathcal{S}$ of points in the polygon is to be guarded (DMSR). We further generalize this version of the problem by associating priorities to the points of $\mathcal{S}$, abbreviated WDMSR. Note that WDMSR may be interesting in situations when not all points are equally important; some need to be guarded more frequently than others. We formulate this idea and show that solving it even in simple polygons is NP-hard. We then propose two approximation algorithms.

2 Preliminaries

The different solutions proposed for the shortest watchman route problem in a simple polygon $\mathbf{P}$ identify a subset of the convex vertices of $\mathbf{P}$ and computes the shortest tour that visits the visibility polygons of these vertices [1, 4]. (The boundary of these visibility polygons are segments interior to $\mathbf{P}$, collinear to boundary edges, called *essential cuts*, that the shortest watchman route needs to intersect.) Indeed, the algorithms work completely independently from how one defines the points to be guarded. When we just want to guard a finite set of points $\mathcal{S} \subset \mathbf{P}$, we can exchange the visibility polygons of the convex polygon vertices for the visibility polygons for the points in

Figure 1: Difference between the shortest watchman route and minimum surveillance tour.

$\mathcal{S}$ and apply the algorithms with these changes. The running time in this case becomes $O(|\mathcal{S}|^3 n \log n)$.

Let T be a polygonal path or a closed polygonal tour in $\mathbf{P}$. We denote the *length* of T by $||T||$.

For a point $p \in \mathbf{P}$ and a closed polygonal tour T, let $\mathcal{H}_T(p) \stackrel{def}{=} \{T \setminus \mathbf{V}(p)\}$, where $\mathbf{V}(p)$ is the visibility polygon of p in $\mathbf{P}$. Assuming that $\mathbf{V}(p)$ intersects T, the set $\mathcal{H}_T(p)$ consists of disjoint subpaths of T, the *hidden pieces* from p. We define

$$hc_T(p) \stackrel{def}{=} \begin{cases} \infty & \text{if } T \cap \mathbf{V}(p) = \emptyset, \\ \max_{X \in \mathcal{H}_T(p)} \{||X||\} & \text{if } T \cap \mathbf{V}(p) \neq \emptyset \end{cases}$$

to be the *hiding cost* of T with respect to p. For MSR and DMSR, given a finite set of points $\mathcal{S}$ in $\mathbf{P}$, respectively, we define the *surveillance cost*, also known as the *delay*, in each case as

$$d(T) \stackrel{def}{=} \max_{p \in \mathbf{P}} \{hc_T(p)\} \quad \text{and} \quad d(T) \stackrel{def}{=} \max_{p \in \mathcal{S}} \{hc_T(p)\}.$$

3 Surveillance Routes vs. Watchman Routes

The counterexample in Figure 1 (due to Langetepe) shows that a shortest watchman route and a minimum surveillance route are not necessarily the same. With appropriate scaling, the length of the shortest watchman route, the dotted green triangle in Figure 1(a), has length $3\sqrt{3} \approx 5.19615$ and the delay is the same value. The tour in Figure 1(b), the dotted blue hexagon, has length 6 but only delay 5, since every point in the polygon sees at least one unit length of the tour. We next prove the following theorem.

Theorem 1 *Let W be a shortest watchman route in $\mathbf{P}$. The tour W is a 2-approximation for MSR in $\mathbf{P}$.*

Proof. Let T be an optimal solution for MSR in $\mathbf{P}$. Pick a point on T and follow T in counterclockwise order until the traced path has visited the visibility polygons of each convex vertex of $\mathbf{P}$. Let v be the last convex vertex seen by the traced path and let p be the earliest point on T that sees v. Follow T backwards from p until v is seen again at p'. Let X be the subpath of T as we go in counterclockwise order from p' to p. The path X visits the visibility polygons of each convex vertex of $\mathbf{P}$, starting at $p' \in \mathbf{V}(v)$ and ending at $p \in \mathbf{V}(v)$ but no other point of X except these endpoints intersects $\mathbf{V}(v)$. Thus, $d(T) \geq ||X||$ and following X from p' to p and back forms a watchman route. Hence,

$$d(W) \leq ||W|| \leq 2||X|| \leq 2d(T). \qquad \square$$

A simple modification of the above proof allows us to extend the relationship between the discrete variants of the shortest watchman route and minimum surveillance route problems in a simple polygon $\mathbf{P}$, given a finite set $\mathcal{S}$ of points to guard.

4 Weighted Discrete Surveillance Routes

To associate weights (or priorities) to the given points of $\mathcal{S}$, we modify DMSR as follows. Let $\mathbf{P}$ be a simple polygon and let $\mathcal{S}$ be a finite set of points inside $\mathbf{P}$. To each point $p \in \mathcal{S}$ is associated a weight $w(p)$. The idea is that points with higher weights have higher priority and need to be guarded more often than ones with lower weights. Given some tour T, we define the *weighted delay* as

$$d_w(T) \stackrel{def}{=} \max_{p \in \mathcal{S}} \{w(p) \cdot hc_T(p)\}.$$

We are ready to formulate WDMSR.

Definition 1 WDMSR: *Given a simple polygon $\mathbf{P}$, a finite set $\mathcal{S}$ of points inside $\mathbf{P}$, and a weight function w on the points in $\mathcal{S}$, find a tour T such that $d_w(T)$ is minimized.*

For simplicity we assume that all weights are positive and that the smallest weight is equal to 1.

4.1 Hardness of WDMSR

The Integer Partition problem is defined as follows.

> INPUT: A finite set $\mathcal{Z}$ of positive integers.
> QUESTION: Is there a subset $\mathcal{Z}' \subseteq \mathcal{Z}$ such that $\sum_{a \in \mathcal{Z}'} a = \sum_{a \in \mathcal{Z} \setminus \mathcal{Z}'} a$?

Since Integer Partition is NP-complete we have the following theorem.

Theorem 2 *WDMSR is NP-hard.*

Proof. We show a reduction from the Integer Partition problem as illustrated in Figure 2. Given a set $\mathcal{Z}$ of positive integers, we construct a polygon $\mathbf{P}$ consisting of a thin horizontal corridor of width $\epsilon \ll 1$ with $|\mathcal{Z}|+1$ *legs*, $L_0, \ldots, L_{|\mathcal{Z}|}$, attached to it. Each leg also has width ϵ and has a pocket at its end; see Figure 2. Leg L_i, $1 \leq i \leq |\mathcal{Z}|$ has length $a_i/2$ from the corridor to the pocket, where a_i is the i^{th} integer in $\mathcal{Z}$. Leg L_0 has length $1/12 + 2\epsilon$ and is directed upward. We further assume that the values $a_i \in \mathcal{Z}$ are sorted in non-decreasing order.

The points of $\mathcal{S}$ are the convex vertices adjacent to the horizontal edges of each pocket with $p_i \in \mathcal{S}$ in L_i. Point p_0 has weight $w(p_0) = 2$, marked blue in Figure 2 and the remaining points have weight $w(p_i) = 1$, for $1 \leq i \leq |\mathcal{Z}|$. The visibility polygons $\mathbf{V}(p_i)$ of the points in $\mathcal{S}$ are the regions marked red and blue in Figure 2.

We compress this structure horizontally so that its width is at most $1/6 + 2\epsilon$ and let $\epsilon \leq 1/(18|\mathcal{Z}|)$. The polygon consists of $5(|\mathcal{Z}| + 1)$ vertices.

Let $Z = \sum_{a \in \mathcal{Z}} a$ and we next prove that $\mathcal{Z}$ can be partitioned into two equal sets if and only if there is a tour T in $\mathbf{P}$ that sees the points of $\mathcal{S}$ and has delay at most $Z + 1$. We prove the two directions of the equivalence separately.

$\Rightarrow$ Suppose that $\mathcal{Z}$ can be partitioned into two sets of equal magnitude, $\mathcal{Z}_1$ and $\mathcal{Z}_2$. We construct a tour T as follows. We start from the visibility polygon $\mathbf{V}(p_0)$ and visit the visibility polygons of the points in $\mathcal{Z}_1$. We then visit $\mathbf{V}(p_0)$ again, followed by the visibility polygons corresponding to points in $\mathcal{Z}_2$. Finally, we go back to $\mathbf{V}(p_0)$.

The hiding cost for p_0 is at most $Z/2 + 2(1/6 + 1/12) = Z/2 + 1/2$. For any other point p_i, the hiding cost is at most $2(Z/2 + 1/3 + 1/6)$. The weighted delay for T is thus

$$d_w(T) = \max \left\{ \begin{array}{ll} w(p_0) \cdot (Z/2 + 1/2), & \\ w(p_i) \cdot (Z + 1) & \mid 1 \leq i \leq |\mathcal{Z}| \end{array} \right\} \leq Z + 1,$$

since $w(p_0) = 2$ and $w(p_i) = 1$ for $1 \leq i \leq |\mathcal{Z}|$.

$\Leftarrow$ Suppose there is a tour T with $d_w(T) \leq Z + 1$. By the same argument as in the proof of Theorem 1, there exists some index j such that as T leaves $\mathbf{V}(p_j)$ it visits each visibility polygon of the remaining points in $\mathcal{S}$ at least once before returning to $\mathbf{V}(p_j)$. We call the trace that T follows from $\mathbf{V}(p_j)$ until $\mathbf{V}(p_j)$ is reached again a *cycle* of T. We first claim that $\mathbf{V}(p_0)$ is visited at least twice during a cycle, but this must hold, otherwise the hiding cost of p_0 is at least Z and therefore $d_w(T) \geq 2Z > Z + 1$, giving us a contradiction. This also means that $j > 0$.

If $\mathbf{V}(p_0)$ is visited at least three times during a cycle, we call each subpath between successive visits at $\mathbf{V}(p_0)$ a *loop* and we have at least two loops completely contained in a cycle. We again have a contradiction since at least one of the loops must have horizontal width at least $1/6$ to visit the rightmost leg, and at least one of the remaining loops must have horizontal width at least $1/12$, otherwise p_0 has hiding cost at least $Z/2 + 1$ (and therefore $d_w(T) \geq Z + 2$) since the previous loop then must visit all the $|\mathcal{Z}|/2$ legs with the longest lengths, because the legs are ordered in non-decreasing order. Any remaining loops must have horizontal width at least ϵ. The point p_j thus has hiding cost at least $Z + 2(1/6 + 1/12 + \epsilon + 3/12) > Z + 1$.

The visibility polygon $\mathbf{V}(p_0)$ is therefore visited exactly twice during a cycle. Consider a subpath X of T between these two successive visits of $\mathbf{V}(p_0)$. Let $X^* = \arg\max\{||X||, ||T \setminus X||\}$. Since $hc_T(p_0) \leq Z/2 + 1/2$, otherwise $d_w(T) > Z + 1$, the length $||X^*|| \leq Z/2 + 1/2$ and we can let $\mathcal{Z}^*$ consist of those integers in $\mathcal{Z}$ corresponding to visibility polygons $\mathbf{V}(p_i)$, $1 \leq i \leq |\mathcal{Z}|$, that X^* visits. The sum $\sum_{a \in \mathcal{Z}^*} a = Z/2$ since all values in $\mathcal{Z}^*$ are integral. By the same token $\sum_{a \in \mathcal{Z} \setminus \mathcal{Z}^*} a = Z/2$, since $||T \setminus X^*|| \leq ||X^*||$ concluding the proof. □

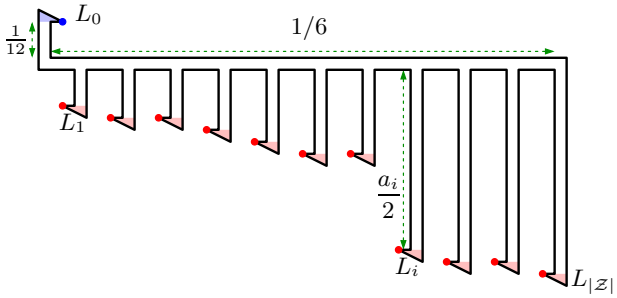


Figure 2: Illustrating the proof of Theorem 2.

We note from the proof above that WDMSR remains NP-hard if the weights are limited to be 1 and 2.

5 Approximations

We start by showing a simple relationship between our two discrete problems.

Theorem 3 *Let W be the shortest tour in $\mathbf{P}$ that sees all points of $\mathcal{S}$. The tour W is a $2w_{\max}$-approximation to WDMSR in $\mathbf{P}$ where $w_{\max}$ is the maximum weight of the points in $\mathcal{S}$.*

Proof. From Theorem 1, the tour W is a 2-approximation for WDMSR when all the weights of the points are 1. If we change the weight of the point with the longest hiding cost to $w_{\max}$, we get an upper bound on the cost of the solution for the maximum weight of $w_{\max}$. It follows that the maximum weight when using W is at most 2 times this weight and thus a $2w_{\max}$-approximation. □

5.1 Two Weight Values

We study the case of WDMSR where points have one of two possible associated weight values, 1 and $w > 1$. From Theorem 2, we know that this restricted case is NP-hard.

We abuse language somewhat and say that a tour *visits* a point $p \in \mathcal{S}$, when we actually mean that the tour intersects $\mathbf{V}(p)$. A point $p \in \mathcal{S}$ is called *low* if $w(p) = 1$ and *high* if $w(p) = w > 1$.

Let W be the shortest tour that visits all points in $\mathcal{S}$, let W_1 be the shortest tour that visits all low points in $\mathcal{S}$, and let W_w be the shortest tour that visits all the high points in $\mathcal{S}$. Each of these tours can be computed in $O(|\mathcal{S}|^3 n \log n)$ time. Let m be the number of low points in $\mathcal{S}$ and we construct a set of tours $U_0, \ldots, U_m$ that each visits all the points in $\mathcal{S}$ and at least one of them has short delay.

U_m is constructed as follows: let p_x be some high point, follow W_w from a point in $\mathbf{V}(p_x)$ until all high

points have been visited, then move to a low point, go back to visit p_x, follow W_w around again, move to another low point, go back to visit p_x, etc., until all low points have been visited. U_m makes as many rounds around W_w as there are low points. Let d be the length of the path along U_m between $\mathbf{V}(p_x)$ and the furthest visibility polygon of any low point and let $p_z \in \mathcal{S}$ be this low point.

The tour U_k, for $1 \leq k \leq m$, is now constructed as follows: let $D_k = \max\{d, ||W_w||/2, ||W_1||/k\}$ and follow W_w from a point in $\mathbf{V}(p_x)$ along W_w until all high points have been visited, visit p_z and move a distance of at most D_k along W_1 visiting low points before going back to $\mathbf{V}(p_x)$, make a tour of W_w, move to W_1 at the point where we left previously and move at most D_k along W_1 before going back to $\mathbf{V}(p_x)$, repeating until all low points have been visited. For $k = 1$, U_1 makes one tour around W_w and one tour around W_1. Finally, we let $U_0 = W$.

We call a tour *weight separated* if every low point is visited exactly once and it can be partitioned into consecutive subpaths $X_1, \ldots, X_t$, such that X_{2i+1} visits *all* high points exactly once but no low points and X_{2i} visits at least one low point but no high point. Tours $U_1, \ldots, U_m$ are all weight separated.

Theorem 4 *At least one of the tours $U_0, \ldots, U_m$ is a 12-approximation to the two-weight WDMSR problem in a polygon $\mathbf{P}$ given a finite set $\mathcal{S}$ of points to guard.*

Proof (sketch). Let T be an optimal solution for WDMSR in $\mathbf{P}$ that sees all points in $\mathcal{S}$ and has minimum weighted delay. Pick a point on T and follow T in counterclockwise order until the traced path finishing at point q has visited each point in $\mathcal{S}$. Let p_j be the last point of $\mathcal{S}$ seen by the trace. Follow T again from q in clockwise order until p_j is visited again at q' and denote this subpath by X. If $w(p_j) = w$, then $d_w(U_0) \leq 2d_w(T)$, since $d_w(T) \geq ||X|| \geq ||W||/2 \geq d_w(U_0)/2$ and the path X sees all points of $\mathcal{S}$. Also, if $||W||/||W_w|| \leq 6$, then $d_w(U_0) \leq 12d_w(T)$. We therefore assume from now on that $w(p_j) = 1$ and that $||W_w|| < ||W||/6$.

Let $p_{j'}$ be the last point in $\mathcal{S}$ visited, moving on T from q towards q' in clockwise order, strictly before q' and let q'' be the first point of encounter with $\mathbf{V}(p_{j'})$ (on this occasion, $p_{j'}$ may have been repeatedly visited before). Let Y be this subpath of T from q to q'' and let T' be the tour obtained by following Y from q to q'' and back to q along the same path. We ensure that T' does not make more than one detour per low point by shortcutting any repetitions if possible. Since $||Y|| \leq hc_T(p_j) \leq d_w(T)$, we have $hc_{T'}(p_j) \leq 2||Y|| \leq 2d_w(T)$ and p_j is visited only once on T' so this hiding cost holds for every low point. Let K be the fewest number of repetitions

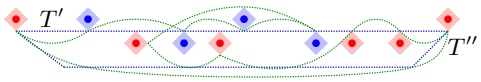


Figure 3: Illustrating the proof of Theorem 4.

of any high points along T'. Again, by shortcutting any repetitions of high points, if possible, we ensure that every high point is visited exactly K times by T' maintaining $d_w(T') \leq 2d_w(T)$ and that each of the K sequences of high points makes a detour to visit each high point at most once.

We transform T' to a weight separated tour T'' as follows: find the shortest subpath H_1 of T' that visits all high points, then follow T' further from H_1 until a high point is reached again. We let this subpath be L_1. We continue along T' until all high points have been visited again, giving H_2 followed by L_2 and continue subdividing T' into $2K$ subpaths, $H_1, L_1, \ldots, H_K, L_K$, each H_i visiting all the high points and L_i only visiting low points. Follow each path H_i shortcutting detours made to all low points not directly on subpaths between high points, until every high point has been visited, then go back and visit all (unvisited) low points that were shortcutted between the first and last high point and connect to L_i, giving a new path Z_i; see Figure 3 where high points are blue and low points are red. We have $T'' = \bigcup_{1 \leq i \leq K} Z_i$ and $||H_i|| + ||L_i|| \leq ||Z_i|| \leq 3||H_i|| + ||L_i||$.

Choose K^* to be the smallest of the two values K, defined above, and K', the largest integer such that $||W_1||/K' \geq \max\{d, ||W_w||/2\}$. The tour U_{K^*} has hiding cost $||W_w|| + ||W_1||/K^* + d \leq 2||Z_i||$ for each high point and each $1 \leq i \leq K^*$, and hiding cost $K^*||W_w|| + ||W_1|| + K^*d \leq 2\sum_{1 \leq i \leq K^*} ||Z_i|| \leq 2||T''||$ for each low point, giving us $d_w(U_{K^*}) \leq 2d_w(T'')$. Hence, $d_w(U_{K^*}) \leq 12d_w(T)$. □

Acknowledgements

The authors wish to thank Prof. Elmar Langetepe for fruitful initial discussions on these problems.

References

[1] M. Dror, A. Efrat, A. Lubiw, and J. Mitchell. Touring a sequence of polygons. In *Proc. 35th STOC'03*, pages 473–482, 2003.

[2] T. Kamphans and E. Langetepe. Inspecting a set of strips optimally. In *Proc. 11th WADS*, pages 423–434, 2009.

[3] J. O'Rourke. *Art Gallery Theorems and Algorithms*. Oxford University Press, 1987.

[4] X.-H. Tan. Fast computation of shortest watchman routes in simple polygons. *Information Processing Letters*, 77(1):27–33, 2001.

On the Traveling Salesman Problem in Solid Grid Graphs

Sándor P. Fekete * Christian Rieck * Christian Scheffer *

Abstract

We consider the Traveling Salesman Problem in solid grid graphs, whose complexity is one of the long-standing problems in *The Open Problems Project*. We disprove a conjecture that a component-minimal 2-factor, i.e., a minimum cardinality set of disjoint cycles that cover all vertices, would yield an optimal tour. Instead, we show that it is sufficient to find a longest cycle to obtain optimal tours—at least in solid grid graphs that contain a 2-factor.

1 Introduction

The Traveling Salesman Problem (TSP) is one of the classic problems of optimization. Easy to describe but provably hard, it shows up in a wide range of application fields, e.g., planning roadtrips, guiding industrial machines, organizing data or mowing a lawn. Discretizations to grid points are common, so many scenarios deal with *grid graphs*, in which vertices are points in the orthogonal integer grid, and edges connect grid points at unit distance.

We consider the TSP in *solid grid graphs* $G = (V, E)$ that do not have any *holes*, i.e., graphs for which the set $\mathbb{N}^2 \setminus V$ with unit-length connections is connected. In 1997, Umans and Lenhart showed that the Hamiltonian Cycle Problem (HCP) is polynomially solvable in these graphs; the more general problem of finding a shortest tour (for which distances between non-adjacent vertices are induced by shortest-path distances in the graph, corresponding to Manhattan distances) has defied all attempts at resolving its complexity. As Problem #54 in *The Open Problems Project*[1] (TOPP), this belongs to a prominent list of long-standing open problems. The complexity of the related Longest Cycle Problem (LCP) in these graphs is also an open question.

Related Work. Itai et al. [5] showed that the HCP is NP-complete for general grid graphs. Umans and Lenhart [8] proved that the HCP is decidable in polynomial time for solid grid graphs. Their algorithm is based on merging components of an initial 2-factor, which is done by flipping the edge parities of a specific class of alternating cycles between different components. Kunas [6] showed that even if there is no Hamiltonian cycle, the edge-flipping algorithm terminates with a 2-factor consisting of a minimum number of components. She also conjectured that these component-minimal 2-factors can be used to solve the TSP in solid grid graphs, which would be sufficient to resolve Problem #54 of TOPP. Arkin, Fekete and Mitchell [2] showed that grid graphs with n vertices and without local cut vertices allow a tour of length at most $6n/5$; the results by Arora [3] and Mitchell [7] imply the existence of polynomial-time approximation schemes. For other classes of grid graphs, e.g., triangular and hexagonal grid graphs, Arkin et al. [1] proved that the HCP is NP-complete in the general case, but decidable in polynomial time for solid triangular grids. The complexity for solid hexagonal grids is an open problem. Asgharian-Sardroud et al. [4] gave a simple 2/3-approximation algorithm for the LCP in solid grid graphs.

Our Contribution. We show that a component-minimal 2-factor is *not* necessarily a substructure of an optimal tour, disproving a conjecture that would have resolved the long-standing Problem #54 of TOPP. Instead, we show that it is sufficient to find a longest cycle to obtain optimal tours—at least in solid grid graphs that contain a 2-factor. We further give some geometric observations and bounds on the LCP, which lead to a number of interesting questions.

2 Solid Grid Graphs with 2-Factor

Any grid graph with a Hamiltonian cycle contains a 2-factor. As the first step in the edge-flipping algorithm of Umans and Lenhart, its existence is easily checked with matching techniques. Kunas [6] extends this by proving that for this class, the edge-flipping algorithm terminates with a component-minimal 2-factor.

2.1 Component-Minimal 2-Factors

The cells between two components of any 2-factor can be of the different types shown in Figure 1. A simple observation is that a type IV cell induces the occurrence of a type III cell, and that a type III cell can be used to decrease the number of components of a given 2-factor by flipping its edge parities. So it is easy to conclude that in any component-minimal 2-factor,

*Department of Computer Science, TU Braunschweig, Germany. {s.fekete,c.rieck,c.scheffer}@tu-bs.de

[1] http://cs.smith.edu/~orourke/TOPP/

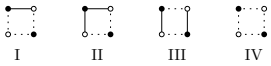


Figure 1: Different types of cells between any two adjacent components of a 2-factor, i.e., not all vertices of these cells belong to the same component. (Vertices are black and white according to their parity in the grid graph, while edges of a 2-factor are shown as solid edges.)

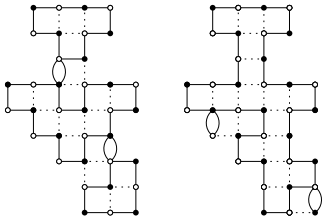


Figure 2: On the left we see a tour that has a component-minimal 2-factor as a substructure. This tour contains four vertices of degree four. On the right, we see that an optimal tour only contains two vertices of degree four.

only cells of type I and type II can occur between its components.

Furthermore, exactly two components are incident to each of these cells. We can construct the dual graph G_k of a component-minimal 2-factor F_k (with k components), where a vertex represents a component and an edge connects two vertices iff their respective components share at least one border cell. Due to the previous observations, it is clear that G_k is a tree.

2.2 First Approach for Using Edge Flips

Kunas suggests the following approach. Compute an initial 2-factor and decrease the number of components, using the edge-flipping algorithm of Umans and Lenhart, until it terminates with a component-minimal 2-factor F_k. Because G_k is a tree, we can connect the components of F_k by doubled edges in a tree-like manner to obtain a tour of length $|V| + 2\,(k-1)$.

Lemma 1 (see [6]) *Let F_k be a component-minimal 2-factor in any solid grid graph G. There exists a tour T in G of length at most $|V| + 2\,(k-1)$.*

This is an upper bound on the length of any tour. Kunas conjectured that it is also a lower bound: We insert additional edges if and only if we cannot merge another two components, so this approach produces—due to this specific substructure—a minimum number of vertices of degree four. It is easy to see that this yields an optimal solution for any solid grid graph with a 2-factor consisting of two cycles.

However, this is *not* true in general; even for graphs with a 2-factor with three components.

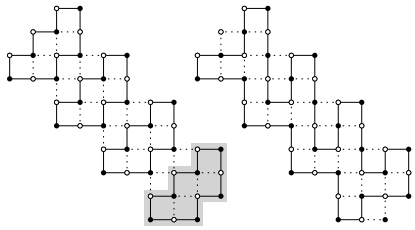


Figure 3: The graph can be extended by adding more W-shaped subgraphs. In the left case we need a doubled edge between any pair of adjacent components, whereas an optimal tour just has length $|V| + 2$.

Theorem 2 *A 2-factor with a minimum number of components is not necessarily a substructure of an optimal tour in solid grid graphs.*

Proof. See the graph in Figure 2. □

The gap between the length of a tour constructed by the previous approach to any optimal tour can get arbitrarily large. An example is given in Figure 3.

2.3 What Optimal Tours Look Like

As shown in Theorem 2, a 2-factor is not necessarily a substructure of optimal tours in solid grid graphs. In the following we work out some important properties of optimal tours. Putting them together establishes the following theorem.

Theorem 3 *Given a solid grid graph that contains a 2-factor. Then there is an optimal tour such that the longest cycle is a substructure of this tour.*

Lemma 4 *If an edge e is used more than twice in a tour T, then there is a shorter tour T', and e is used at most twice.*

The high-level idea is the following. Assume that such an edge e is used an even number of times. Then by removing e and all copies of e, the degree of both incident vertices remains even; we either obtain a tour or we just have to add e twice, which yields a shorter tour in both cases. A similar argument holds for any edge that is used an odd number of times. This directly implies that there is always a tour in which all vertices have degree at most eight.

Lemma 5 *In every solid grid graph that contains a 2-factor, an optimal tour cannot contain vertices of degree eight.*

Proof. We prove this by contradiction to the optimality of the tour. Due to space constraints we only show one case; the proof can be completed by similar arguments. As a consequence of Lemma 4, a vertex v of degree eight must be connected to each of its four neighbors by doubled edges. If v is no cut vertex,

Figure 4: The modification from left to right yields a shorter tour, because the degree of the middle vertex decreases by two.

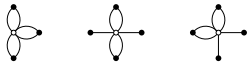


Figure 5: The different edge configurations of a degree-six vertex.

then at least three vertices of its 8-neighborhood must exist. Figure 4 shows one such possible edge configuration. One can see that the transformation from left to right yields a tour with shorter length. □

Lemma 6 *Every solid grid graph with a 2-factor has an optimal tour in which no vertex has degree six.*

Proof. This can be proven by similar arguments as in Lemma 5, checking the different configurations for a degree-six vertex (see Figure 5) in its 8-neighborhood. There are situations in which a degree-six vertex can occur in an optimal tour, but there are local modifications to this tour that move a doubled edge incident to this vertex, creating vertices of degree two and four. Details are omitted due to limited space. □

Figure 6: The two different types of a degree-four vertex. The dotted cycle in the right shows that the black vertex can have degree two or four. The left one is called a *cross vertex*, whereas the right one is an *isolated vertex*, iff the black vertex has degree two.

In order to prove Theorem 3, we have to show that we can modify the different types of degree-four vertices in an optimal tour (see Figure 6), such that we get a longest cycle. First consider the case in which two subtours are connected by a doubled edge. Without loss of generality, let the boundary between these subtours be neither a bridge nor a cut vertex. Then we can easily merge both subtours to get a single cycle that contains all but two vertices of the initial subtours. There is only one kind of unmergeable component, which is shown in Figure 7.

Lemma 7 *Let $G = (V, E)$ be solid grid graph without cut vertices and let F_k be a component-minimal 2-factor in G. Then there is a cycle of length at least $|V| - 2\,(k - 1 + \sigma)$, where σ denotes the number of unmergeable components.*

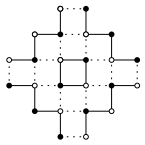


Figure 7: If all but the vertices of the nested C_4 are connected in a single cycle, then we cannot extend this cycle by some of these inner vertices.

In general, given an arbitrary 2-factor, this cycle is *not* a longest cycle (see Figure 3). However, if a doubled edge connects two cycles in an optimal tour, we can merge them into a single one—containing two isolated vertices. Now we only have to show that the *cross-shaped* vertex of degree four can be modified, such that we get a doubled edge between a single vertex and a bigger cycle, i.e., an isolated vertex.

Lemma 8 *Let G be solid grid graph without cut vertices that contains a 2-factor. In an optimal tour of G, a cross vertex is equivalent to an isolated vertex.*

Proof. See Figure 8 for the main idea; details are omitted due to limited space. □

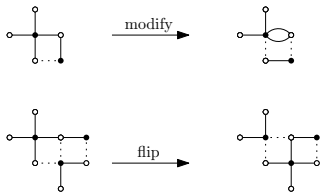


Figure 8: These are basically all possible configurations in the extended neighborhood of a cross vertex. Preserving the tour, we can either directly modify a cross vertex to an isolated one (Top), or we can *flip* it to obtain the upper configuration (Bottom).

2.4 Longest Cycles

Knowing that longest cycles yield optimal tours in solid grid graphs with a 2-factor, makes it interesting to exploit the structure of component-minimal 2-factors F_k. Recalling Figure 3 and Lemma 7, it is easy to argue that an upper bound for any cycle is $|V| - \lambda$, where λ denotes the number of leaves in G_k. It is plausible to conjecture that λ could also denote the number of odd vertices in G_k. Unfortunately, this is not true in general.

Instead, we analyze the color of the vertices that are left in the leaf-corresponding vertex sets to improve this bound. To this end, we count the leaves in G_k for which the corresponding component in F_k has black vertices adjacent to another component, and subtract one for each leaf in which these vertices are white. Let ζ denotes the absolute value of this sum, then $|V| - (\lambda + \zeta)$ is an upper bound for any cycle in G. The intuitive idea is that the number of vertices left

in the leaf-corresponding sets does not suffice for connecting all vertices of the inner components; this is analogous to the Tutte-Berge formula from matching theory. Unfortunately, it is not easy to improve this bound because there are no specific structures that can be used to achieve this, i.e., some cutting sets of size two or the like (see Figure 9).

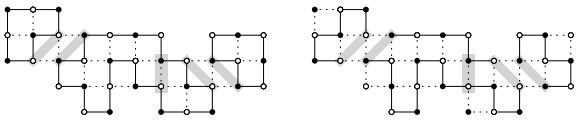


Figure 9: The upper bound of $|V| - (\lambda + \zeta)$ on any cycle results in $|V| - (2 + 0)$. Analyzing the shaded cutting sets yields an upper bound of $|V| - 4$ on any cycle in this graph.

3 General Solid Grid Graphs

A general solid grid graph can be decomposed into (A) vertex-clusters that contain a 2-factor, (B) clusters that are 2-connected, (C) 1-dimensional paths that connect different clusters or single vertices, and (D) clusters that contain a cut vertex (see Figure 10). It

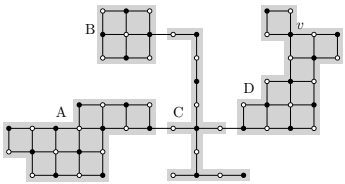


Figure 10: A general solid grid graph with the component classification given in Section 3.

is trivial to observe that all edges on a 1-dimensional path must be used twice in any optimal tour. It can also be shown that the Lemmas 5 and 6 can be adapted by similar arguments to the case that a vertex cluster is 2-connected but does not contain a 2-factor. Finally, it is easy to see that one can decompose a cluster at a cut vertex, solve the problem for each subcluster and merge them afterwards. This implies that many of the previous results can be reused.

Unfortunately, we were not able to extend the arguments of Lemma 7 and Lemma 8 for general 2-connected solid grid graphs. Without these, we were not able to show that a longest cycle is also part of an optimal tour in general solid grid graphs. Nevertheless, we are strongly convinced that this is the case, as stated in the following conjecture.

Conjecture 1 *In general solid grid graphs, a longest cycle is part of an optimal tour.*

4 Future Work

We have shown that a component-minimal 2-factor is not necessarily a substructure of an optimal tour in a solid grid graph with a 2-factor. We were able to prove that for these graphs, a longest cycle is part of an optimal tour. We also gave a number of geometric observations on longest cycles. Some of these results can be adapted to general solid grid graphs. We are convinced that a longest cycle is also part of an optimal tour in general solid grid graphs.

The computational complexity of both the TSP and the LCP in solid grid graphs is still open. It may be useful to know whether there always exists a longest cycle such that all vertices that are not part of this cycle lie on the boundary of the graph. If true, it may be possible to solve the question of how many vertices of the boundary must be deleted, such that the remaining graph is Hamiltonian. Another challenging question is whether there is always a longest cycle for which all unvisited vertices are adjacent to it.

References

[1] E. M. Arkin, S. P. Fekete, K. Islam, H. Meijer, J. S. B. Mitchell, Y. N. Rodríguez, V. Polishchuk, D. Rappaport, and H. Xiao. Not being (super)thin or solid is hard: A study of grid Hamiltonicity. *Comput. Geom.*, 42(6-7):582–605, 2009.

[2] E. M. Arkin, S. P. Fekete, and J. S. B. Mitchell. Approximation algorithms for lawn mowing and milling. *Comput. Geom.*, 17(1-2):25–50, 2000.

[3] S. Arora. Polynomial time approximation schemes for Euclidean Traveling Salesman and other geometric problems. *J. ACM*, 45(5):753–782, 1998.

[4] A. Asgharian-Sardroud and A. Bagheri. An approximation algorithm for the longest cycle problem in solid grid graphs. *Disc. Appl. Math.*, 204:6–12, 2016.

[5] A. Itai, C. H. Papadimitriou, and J. L. Szwarcfiter. Hamilton paths in grid graphs. *SIAM J. Comput.*, 11(4):676–686, 1982.

[6] W. Kunas. Kürzeste Rundreisen auf einfachen Gittergraphen. Diplomarbeit, Technische Universität Braunschweig, 2003.

[7] J. S. B. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric TSP, k-MST, and related problems. *SIAM J. Comput.*, 28(4):1298–1309, 1999.

[8] C. Umans and W. Lenhart. Hamiltonian cycles in solid grid graphs. In *38th Ann. Symp. Found. Comp. Sci., FOCS '97*, pages 496–505, 1997.

Covering Tours with Turn Cost: Variants, Approximation and Practical Solution

Sándor P. Fekete* Dominik Krupke*

Abstract

For a given set P of points in the plane, the Angular-Metric Traveling Salesman Problem (AM-TSP) asks for a tour on P that minimizes the total turn along the tour. While there exists a PTAS for the Euclidean TSP, for the AM-TSP only a $O(\log n)$ approximation algorithm is known. We introduce a number of generalizations and provide approximation algorithms whose performance depend on the angular resolution. We also develop exact methods for computing provably optimal solutions, and present an array of experimental results.

1 Introduction

Consider an outdoor setting with a number of obstacles. Swarms of mosquitos populate the area, with a number of known hotspots. How can we lower the danger of diseases by zapping the mosquitos with a flying drone, such as the one shown in Figure 1?

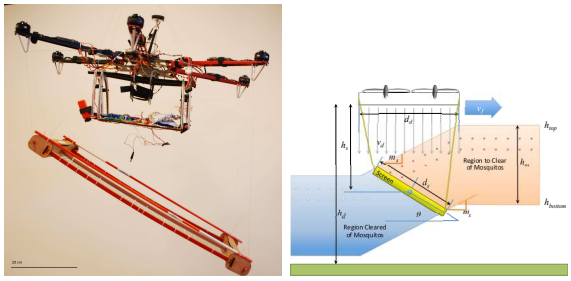


Figure 1: A drone equipped with an electrical lattice to hunt mosquitoes. Images by Aaron Becker.

Visiting a set of points by an optimal tour is a natural and important problem, both in theory and practice. If we are only concerned with minimizing the total distance traveled for visiting all points this is the classic Traveling Salesman Problem (TSP). However, for path planning by a flying robot, we also incur a cost for changing direction; this is related to the *Angular-Metric TSP* (AM-TSP), in which the objective is to minimize the total turn. In addition, we may want to focus on a subset of the points in order to provide better coverage, incurring a penalty for the uncovered ones.

*Department of Computer Science, TU Braunschweig, Germany. s.fekete@tu-bs.de, krupke@ibr.cs.tu-bs.de

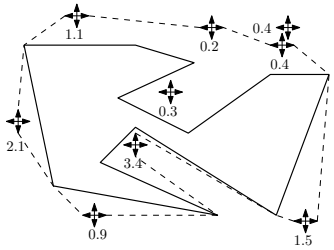


Figure 2: An example instance with obstacles and individual penalty costs for the points in P.

Related Work. Angle-restriced tour problems were studied by Fekete and Woeginger [4]. Touring points in the plane with minimal turn cost was considered by Aggarwal et al. [1], who provide a $O(\log n)$ approximation algorithm, but also show that already the cycle cover version is NP-hard. Arkin et al. [3] consider different grid-based versions of covering with turn cost, and provide a spectrum of approximation algorithms. Minimizing the total turn cost can be modeled as a special case of the quadratic TSP, which has received a fair amount of attention; see [5, 6, 9, 8] for research in optimal solutions and heuristics.

Our Results. We consider a number of variants for AM-TSP, motivated by practical applications. In particular, we consider the setting in which the set of possible headings at visited points is discretized; this also allows it to easily add polygonal obstacles into the environment. We also provide approximation algorithms for the generalization in which a penalty can be paid instead of covering a point. In addition, we present computational results.

2 Preliminaries and Problems

Preliminaries. We are given a set of points $P \subset \mathbb{R}^2$ in a polygonal environment with the obstacles $\mathcal{O}$. For every point $p \in P$ a set of ω angles $\delta(p) \in [0, \pi)^\omega$ that describe possible headings when covering p; each angle corresponds to two possible, opposite headings. A *pose* consists of a position and a heading. The respective set of poses results in an undirected weighted graph. There is weighted edge between any two poses that represents the cheapest collision-free polygonal path connecting them. The travel cost is a linear combination of the sum of turn angles (with coefficient τ) and the length (with coefficient κ).

Problems. The *Full Cycle Cover Problem* (FCCP) asks for a set of non-trivial cycles of minimum total cost, such that every point is covered. The *Full Tour Problem* (FTP) asks for a single cycle of minimum total cost that covers all points. The *Penalty Cycle Cover Problem* (PCCP) and the *Penalty Tour Problem* (PTP) are defined analogously, but points may be left uncovered by paying an individual penalty $\rho(p) \in \mathbb{R}_0^+$ for every omitted point $p \in P$.

All problem variants are NP-hard. The hardness of the Full Tour Problem is implied by the hardness of the Euclidean TSP; the Full Cycle Cover Problem is NP-hard with $\omega \geq 2$ by a straightforward adaption of the NP-hardness proof for the angular metric cycle cover problem by Aggarwal et al. [1]. Clearly, this implies hardness for the penalty variants. For $\omega = 1$, the problem can be solved using a minimum weight perfect matching on an appropriate auxiliary graph.

A subproblem is to calculate the cheapest transition between two poses around polygonal obstacles. If there are only distance costs, this problem equals the Euclidean shortest path, for which only the visibilty graph needs to be considered. It can easily be shown that this is also true with turn costs and the graph can easily be transformed such that the turn costs are integrated in edge weights resulting in a complexity of $O(|V_\mathcal{O}|^2 * \log |V_\mathcal{O}|)$ for the calculation of a cheapest transition, where $V_\mathcal{O}$ are the vertices of the obstacles.

3 Approximation Algorithms

We now propose approximation techniques for all four problems. Due to limited space, we only outline the main ideas; details are left to the full paper.

Note that for all the proposed approximation algorithms, the approximation factor and the runtime both depend linearly on the maximum number of orientations ω, which is assumed to be constant.

3.1 Full Coverage

Theorem 1 *For a fixed ω, there is a $2 * \omega$ approximation algorithm for the FCCP. In case $\omega = 1$, the solution is optimal.*

Proof. Solve the LP-relaxation of the integer program (IP); select for each point the orientation of highest variable value. Do a minimum weight perfect matching on the vertices associated with these points. Both takes polynomial time, with the LP-relaxation being the dominant part.

Because every point has at most ω orientations, at least one orientation of each point is used with a fractional weight of at least $1/\omega$. Multiplying the solution by ω and applying some local modifications that do not increase the cost (like skipping a point) yields a half-integral solution. This can be multiplied by two to obtain an integral solution. By further local modifications that do not increase the cost (possibly even decrease it), we obtain a perfect matching with at most $2 * \omega$ times the cost of the LP-relaxation. This (not minimal) perfect matching is an upper bound. □

To connect the cycles provided by Theorem 1, we simply use a minimum spanning tree (MST). Doubling the edges results in cycles with u-turns on the original cycles, which can be connected with no additional cost (but the u-turns from the doubling involve an extra cost).

Theorem 2 *For a fixed ω, there is a $4 * \omega + 2$ approximation algorithm for the FTP. In case of $\omega = 1$, there is a 4-approximation algorithm.*

Proof. We compute a $2 * \omega$-approximation of the cycle cover using Theorem 1, then connect these cycles via an MST. The MST provides $m - 1$ edges for a cycle cover with m cycles. An edge between two cycles corresponds to the cheapest connection between two of it points ($\in P$), ignoring the headings at the end. These $m - 1$ edges are doubled to create a valid tour. The minimum spanning tree is a lower bound on the optimal value. Connecting the edges to the cycles involves additional turn costs (360° for each doubled edge), but these can be charged to the cycles, because every cycle has a turn angle sum of at least 360°. □

3.2 Penalty Coverage

The adaption of the approximation algorithms to the penalty variants is surprisingly simple.

Theorem 3 *For a fixed ω, there is a $2 * (\omega + 1)$ approximation algorithm for a PCCP.*

Proof. We proceed as in Theorem 1, but we add an artificial orientation that allows a single artificial cycle with the cost of the penalty. □

From this penalty cycle cover we use the prize-collecting Steiner tree to select and connect good cycles. The connecting of the selected cycles via the edges of the tree is identical to full coverage. Only the analysis is slightly more difficult.

Theorem 4 *For a fixed ω, there is a $4 * (\omega + 1) + 4$ approximation algorithm for the PTP.*

Proof. We first compute a penalty cycle cover approximation with a factor of $2 * (\omega + 1)$, using Theorem 3. We remove all points for which the penalty in the cycle cover has been paid. Next we compute a 2-approximation of the prize-collecting Steiner tree, using the approximation algorithm of Goemans and Williamson [7] that has a time complexity of $O(n^2 \log n)$. This is done on a graph that contains

all remaining points, with edge costs corresponding to shortest paths with turn costs but arbitrary start and goal headings. For two points that are in the same cycle, we set the cost to zero. We remove all cycles for which no point is in the resulting prize-collecting Steiner tree. All other components we connect by selecting edges from the tree (i.e., $m-1$ edges for m cycles). This can be done by iterating over all edges in the tree, adding an edge if it connects two different components. Clearly, the cost of an optimal prize-collecting Steiner tree is a lower bound for the tour. Due to the 2-approximation, the sum of all edge weights and penalties is at most twice the cost of the optimal penalty tour. The selected edges are transformed to cycles by doubling them and adding 180° turns at the ends. We can merge the cycles with no additional cost, as in Theorem 2. This results in at most four times the cost of the optimal tour plus $2 * (m-1) \times 180°$ turns for m cycles in the cycle cover. As every cycle in the cycle cover has also at least 360°, we can charge the 180° turns to the cycles, which leads to $2 * 2 * (\omega + 1) * \mathrm{OPT}$. Combined, this results in $2 * 2 * (\omega + 1) * \mathrm{OPT} + 2 * 2 * \mathrm{OPT} = (4 * (\omega + 1) + 4) * \mathrm{OPT}$. □

4 Integer Programming

We work on an auxiliary graph $G(V, E)$: For every point $p \in P$ with orientations $\delta(p)$, we create the vertices $V(p) = \bigcup_{\alpha \in \delta(p)} \{v_{p,\alpha}, v_{p,\alpha+\pi}\}$ representing the two poses by which a point can be left/entered through one of its orientations. Furthermore, there is an edge $e = \{v, v'\}$ between any two $v = v_{p,\alpha} \in V(p)$ and $v' = v_{p',\alpha'} \in V(p')$ with the cost $c(e)$ representing the minimum cost path from the pose of being at p and heading α to the pose of being at p' and heading $\alpha' + \pi$ (this cost is symmetric). Entering on the vertex for the pose of being at p' and heading $\alpha' + \pi$ implies leaving through the vertex for the pose of being at p' with the heading α'.

For the cycle cover variant, there is also an edge $e = \{v_{p,\alpha}, v_{p,\alpha+\pi}\}$ for every $p \in P$ with the cost of the cheapest cycle (covering it and at least one other point). cycle cover, points can be in two unconnected cycles even in an optimal solution. The additional edge is used to implicitly represent this kind of cycle.

The integer programming formulation for cycle cover can be given as follows:

$$\min \quad \sum_{e \in E} c(e) * x_e \tag{1}$$

$$\text{s.t.} \quad \sum_{e \in E(v_{p,\alpha})} x_e = \sum_{e \in E(v_{p,\alpha+\pi})} x_e \quad \begin{matrix} \forall p \in P \\ \forall \alpha \in \delta(v) \end{matrix} \tag{2}$$

$$\sum_{\alpha \in \delta(p)} \sum_{e \in E(v_{p,\alpha})} x_e = 1 \qquad \forall p \in P \tag{3}$$

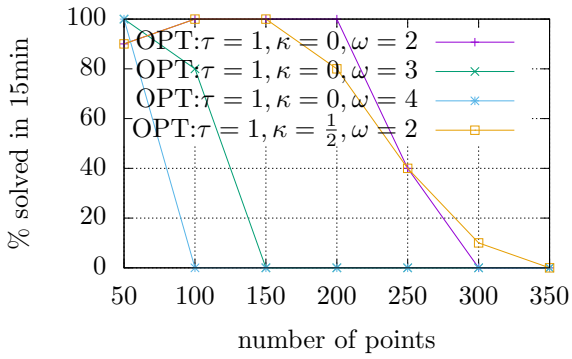


Figure 3: Percentage of tour instances solved to optimum within 15 min. 10 instances for each size $50, 100, \ldots, 350$. The cycle cover variant is only slightly better. i

$$x_e \in \{0, 1\} \quad \forall e \in E \tag{4}$$

Eq. 2 states that if and only if there is an incoming edge on one side, there has to be an outgoing edge on the opposite site. Eq. 3 states that there have to be exactly two edges entering/leaving (using the symmetry induced by the previous equation). $E(v)$ represents the set of edges incident to v.

The subcycle elimination constraints for obtaining a tour can be adapted directly from the TSP.

$$\sum_{e \in E(V(C), V(P \setminus C))} x_e \geq 2 \quad \forall C \subsetneq P, C \neq \emptyset \tag{5}$$

Hence, the IP for tours is given by adding Eq. 5 to the cycle cover formulation. The penalty versions are omitted due to space constraints.

5 Experiments

For the full coverage problem variants, we implemented the integer programs and the approximation algorithms for tour and cycle cover without obstacles. In this section we discuss the experimental results for these implementations. Experiments were executed on modern desktop computers equipped with an *Intel(R) Core(TM) i7-6700K CPU @* 4.00 GHz and 64 GB of RAM. The used CPLEX version was 12.5.0.0 with the parameters `EpInt`=0, `EpGap`=0, `EpOpt`=1×10^{-9}, and `EpAGap`=0. No further optimizations were performed. As cost parameters we used $\tau = 1, \kappa = 0$, with one additional run with $\omega = 2, \tau = 1, \kappa = 0.5$. Experiments were run for 10 random instances in the unit square per size $50, 100, \ldots, 350$. As passing orientations $\{i * \frac{\pi}{\omega} | i = 0, \ldots, \omega - 1\}$ were chosen for all points equally.

Fig. 3 shows the percentage of instances solved to optimum within 15 min via the integer program for tours. The cycle cover variant is only slightly better. It can be seen that we can solve 50% of the instance of

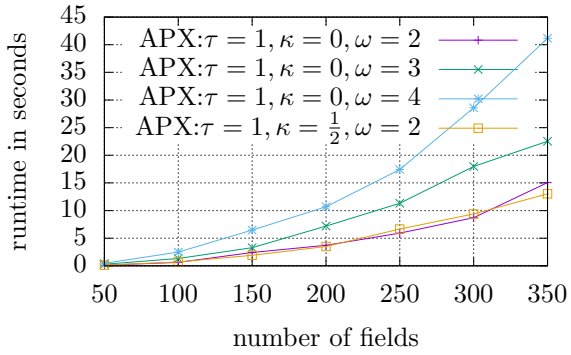


Figure 4: Average runtime of the approximation algorithm for cycle cover and tour (with nearly identical runtime). For each size $50, 100, \ldots, 350$,10 instances were tested.

size up to 200 points for $\omega = 2$. Already for $\omega = 3$ the performance drops strongly such that the maximum instance size becomes 100. For $\omega = 4$ only the smallest instances (50 points) have been solved in time. Hence, with our formulation only for $\omega = 2$ there is a serious advantage compared with the angular metric cycle cover and traveling salesman problem (using the work of Aichholzer et al. [2] as reference point).

The average runtime of the approximation algorithm is shown in Fig. 4. Here we do not differentiate between cycle cover and tour because they have nearly the same runtime. It can be seen that ω also has a lot of influence on the runtime of the approximation algorithm and the runtime for $\omega = 4$ grows significantly stronger than for $\omega = 2$. However, the runtime is much shorter than for the integer programs. The run with $\omega = 2, \tau = 1, \kappa = 0.5$ is nearly identical to the run with $\omega = 2, \tau = 1, \kappa = 0$

The objective value of the approximation algorithm for cycle cover differs only slightly from the optimum. For $\omega = 2$ the difference is on average less than 5%, with a maximum difference of 7.3%. The differences are higher for $\omega = 3$ and $\omega = 4$, with a maximum difference of 17.8%, but there are too few instances solved for a reliable statement. The difference is higher for the tour version, but still close to the optimum and far better than the proven bounds. The maximum difference found is 1.485 times the optimal value for $\omega = 4$. It has to be noted that the implementation does not do a local optimization of the cycle connections, i.e., the points of the connection are visited multiple times. A removal of the redundant visits could further improve the ratio.

We further considered random instances with $\omega = 2$, $\tau = 1$, and $\kappa = 0$ up to a size of 2000 points. For instances with 1700 points and more, the memory consumption of the linear program becomes problematic. The average runtime at this point is roughly 8 min.

6 Conclusion

The assumption that the discretization makes the angular metric traveling salesman problem simpler is only partially true. Surprisingly, the integer program becomes very slow already for low resolutions, which is also true for the cycle cover variant. Experiments for the penalty variants are still to be performed. The approximation algorithm is only practical for very low resolutions. The experimental quality of the solutions of the approximation algorithm is considerably better than the worst-case ratio, but this evaluation is only based on few and relatively small instances. It would also be interesting to evaluate the quality of the solutions compared to the optimal solutions of the AM-TSP. The bottleneck of the approximation algorithm is solving the linear program, but often there are possibilities of replacing the blackbox LP-solver by a more direct approach.

References

[1] A. Aggarwal, D. Coppersmith, S. Khanna, R. Motwani, and B. Schieber. The angular-metric traveling salesman problem. *SIAM Journal on Computing*, 29(3):697–711, 2000.

[2] O. Aichholzer, A. Fischer, F. Fischer, J. F. Meier, U. Pferschy, A. Pilz, and R. Stanek. Minimization and maximization versions of the quadratic traveling salesman problem. Preprint of the Institute for Numerical and Applied Mathematics of the University of Goettingen, Mar. 2016.

[3] E. M. Arkin, M. A. Bender, E. D. Demaine, S. P. Fekete, J. S. Mitchell, and S. Sethia. Optimal covering tours with turn costs. *SIAM Journal on Computing*, 35(3):531–566, 2005.

[4] S. P. Fekete and G. J. Woeginger. Angle-restricted tours in the plane. *Computational Geometry*, 8(4):195–218, 1997.

[5] A. Fischer, F. Fischer, G. Jäger, J. Keilwagen, P. Molitor, and I. Grosse. Exact algorithms and heuristics for the quadratic traveling salesman problem with an application in bioinformatics. *Discrete Applied Mathematics*, 166:97–114, 2014.

[6] A. Fischer and C. Helmberg. The symmetric quadratic traveling salesman problem. *Mathematical Programming*, 142(1-2):205–254, 2013.

[7] M. X. Goemans and D. P. Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24(2):296–317, 1995.

[8] G. Jäger and P. Molitor. Algorithms and experimental study for the traveling salesman problem of second order. In *Combinatorial Optimization and Applications*, pages 211–224. Springer, 2008.

[9] B. Rostami, F. Malucelli, P. Belotti, and S. Gualandi. Quadratic TSP: A lower bounding procedure and a column generation approach. In *Computer Science and Information Systems (FedCSIS), 2013 Federated Conference on*, pages 377–384. IEEE, 2013.

On the Generation of Spiral Paths Within Planar Shapes

Martin Held* Stefan de Lorenzo*

Abstract

We simplify and extend prior work by Held and Spielberger [CAD 2009, CAD&A 2014]: We use a linearization to derive a simple algorithm that computes a spiral path inside of planar shape bounded by straight-line segments and circular arcs. Our spiral paths are continuous and without self-intersection, respect a user-specified maximum step-over distance, start in the interior and end at the boundary of the shape. We also extend this basic algorithm to double spirals that start and end at the boundary.

1 Introduction

Several applications require to cover a planar shape by moving a circular disk along a path. E.g., in machining applications the disk models the cross-section of a tool and the area models a so-called pocket. Similarly, the disk may represent the area covered by a spray nozzle or the area of visibility of a camera device used for aerial surveillance.

Traditional strategies for path generation include zigzag patterns and the use of offset curves to form contour-parallel patterns. Common to these traditional strategies is the fact that the resulting paths contain lots of sharp corners, i.e., abrupt changes of the direction. The higher the speed or the moment of inertia of the moving object represented by the disk is, the more these directional discontinuities become a problem. E.g., for a high speed machining (HSM) application, an abrupt change of direction requires the tool to slow down to near-zero speed, change its direction and then accelerate until the desired maximum speed is reached again.

In order to avoid sharp directional discontinuities, (spiral) paths have been studied. Bieterman and Sandstrom [2] present an approach based on partial differential equations (PDEs) to compute spiral tool paths inside star-shaped pockets. Abrahamsen [1] constructs a polygonal spiral inside a pocket bounded by straight-line segments. Held and Spielberger [4, 5] used the medial axis of the pocket to generate spiral paths for general non-convex pockets with or without islands. The key ingredient of their approach are circles centered on the medial axis whose radii increase as time progresses. Portions of these circles are interpolated and connected by other circular arcs to form a G^1-continuous path.

Since the algorithm by Held and Spielberger [4, 5] is difficult to analyze theoretically and even more difficult to implement reliably, in this work we pick up their overall idea but simplify it significantly: A linearization of the medial axis allows to come up with an algorithm for a raw G^0-continuous spiral path that is easy to implement. (And, indeed, an implementation of this algorithm is already in commercial use at our industrial partner.) The spiral path is continuous, without self-intersection, and respects a user-specified maximum step-over distance. This raw path can then be boosted to G^1-continuity or C^2-continuity by using a (one-sided) approximation by biarcs or cubic B-splines.

As in the work by Held and Spielberger [4, 5], our spiral path starts in the interior of the pocket and ends at its boundary. The simplicity of our approach allows to generalize this scheme and to devise double-spiral paths that start and end at arbitrary points on the boundary. This makes it easier to cover a complex shape by one continuous spiral path by (1) decomposing the shape into simpler sub-areas, (2) computing a (double) spiral within every sub-area, and (3) linking the individual spirals to form one continuous path. While such a double-spiral path is unsuited for machining, it may find use in other applications, such as spray painting, aerial surveillance, or path finding algorithms for rescue missions.

2 Preliminaries

If a disk of radius ρ that moves has to stay within a shape during the entire movement then it is obvious that its center can never get closer to the boundary then ρ, even if this constraint results in some areas of the shape being uncovered. (E.g., at convex corners of the boundary.) The loci of all permissible positions of the center can be obtained as the Minkowski difference of the shape and a disk of radius ρ centered at the origin. We call this set a *pocket*, P. Hence, the area P' to be covered by our moving disk is given by the Minkowski union of P with a disk of radius ρ centered at the origin. It is well-known that (1) the boundary of P also consists of $O(n)$ straight-line segments and circular arcs if the initial shape was bounded by n straight-line segments and circular arcs, and (2) that it can be obtained in $O(n \log n)$ time via

*Universität Salzburg, FB Computerwissenschaften, 5020 Salzburg, Austria.

Voronoi-based offsetting.

We assume that P is path-connected and simply-connected. If P were disconnected then we would run our algorithm separately for every connected component. If P contains islands (i.e., is multiply-connected) then we convert it to a simply-connected area by introducing bridges [5].

It is natural to break a spiral that winds around a point r for k times into a sequence of k individual portions, where each portion corresponds to one full turn around r. We call such a portion of a spiral a *lap*. Then the *step-over distance* at point p of lap ℓ_{i+1} is the minimum distance from p to the next inner lap ℓ_i. It is obvious that, in general, the step-over distance has to be less than the diameter of the disk that is being moved in order to avoid regions of P' that are not covered. In practice, considerably smaller step-over distances are used, though. For HSM, a good step-over value is a fraction of the diameter that depends on the material of the cutter as well as the workpiece. (It is largely independent of the geometry of the pocket.) E.g., for aluminum or (non-hardened) steel a typical maximum step-over is given by about 15% of the diameter. In any case, it is important that the user can specify a maximum step-over Δ that the spiral path has to keep.

3 The Medial Axis Tree

In order to simplify the algorithm by Held and Spielberger [4] we approximate every edge of the medial axis $\mathcal{MA}(P)$ of the pocket P by a polygonal chain. The vertices of such a polygonal chain are obtained by placing uniformly distributed sample points on the edge such that the maximum length of a segment of the chain is less than a user-supplied or heuristically determined value λ. This process yields the discrete medial axis $\mathcal{MA}'(P)$. We refer to the sample points on $\mathcal{MA}(P)$ and the original nodes of $\mathcal{MA}(P)$ as *vertices* of $\mathcal{MA}'(P)$.

As usual, the *clearance*, $\mathrm{clr}(p)$, of a point p on $\mathcal{MA}'(P)$ is the radius of the largest disk centered at p that fits into P. For every vertex p of $\mathcal{MA}'(P)$ we consider the points $p_1, p_2, \ldots, p_k$ where the clearance disk of p touches the boundary ∂P of P, and construct the clearance line segments $\overline{pp_1}, \overline{pp_2}, \ldots, \overline{pp_k}$. (If p happens to be the center of a circular arc a of ∂P then we select finitely many points on a which are uniformly spaced, with a spacing less than λ.)

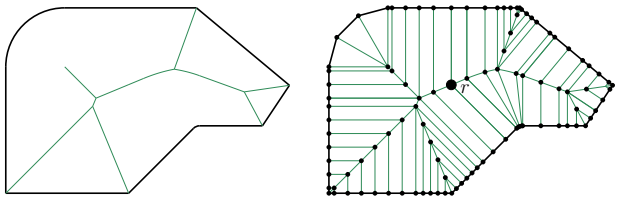
Let $\mathcal{L}$ be the set of all clearance line segments defined by vertices of $\mathcal{MA}'(P)$. We add $\mathcal{L}$ to $\mathcal{MA}'(P)$ and get $\mathcal{MA}''(P) := \mathcal{MA}'(P) \cup \mathcal{L}$.

Both $\mathcal{MA}'(P)$ and $\mathcal{MA}''(P)$ form a tree because P does not contain islands. By choosing one vertex r as root we can turn $\mathcal{MA}''(P)$ into a rooted tree $\mathcal{T}_r(P)$, the *discrete medial axis tree* derived from $\mathcal{MA}''(P)$.

Since all edges of $\mathcal{T}_r(P)$ correspond to line segments, it is easy to compute the (Euclidean) length $d_{\mathcal{T}_r(P)}(p,q)$ of the path between two vertices p, q of $\mathcal{T}_r(P)$. This allows to define the *Euclidean height* of a vertex p of $\mathcal{T}_r(P)$ as

$$h_{\mathcal{T}_r(P)}(p) := \max_q \big(d_{\mathcal{T}_r(P)}(p,q) + \mathrm{clr}(q)\big),$$

where the maximum is taken over all vertices q of the sub-tree of $\mathcal{T}_r(P)$ rooted at p. As in [5] we assume that $\mathcal{T}_r(P)$ is *height-balanced*, that is, that $h_{\mathcal{T}_r(P)}(r)$ is assumed for two different leaves of $\mathcal{T}_r(P)$. (If no such vertex exists then we insert a new vertex within an edge of $\mathcal{T}_r(P)$ in order to achieve a perfect balance.) See Figure 1. (For the sake of visual clarity we show this toy example with a very course discretization and an unrealistically large step-over distance, which result in spiral paths that do not look smooth.)



(a) (b)

Figure 1: (a) Medial axis. (b) Height-balanced discrete medial axis tree $\mathcal{T}_r(P)$.

4 Impulse Propagation

As in [4] we consider an impulse which starts at the root r of the discrete medial axis tree $\mathcal{T}_r(P)$, which is active during the time interval $[0,1]$, and which discharges concurrently at the leaves of $\mathcal{T}_r(P)$. Let e be an edge between the nodes p and q of $\mathcal{T}_r(P)$, where p is the parent node of q inside $\mathcal{T}_r(P)$. The velocity v_e of the impulse at e is given by

$$v_e := \frac{h_{\mathcal{T}_r(P)}(q) + l_e}{1 - t_p},$$

where l_e is the length of e and t_p is the time when the impulse reached p. Since the impulse starts at time $t_r = 0$ at r, we can recursively compute the time when the impulse reaches a specific vertex or any point within an edge of $\mathcal{T}_r(P)$.

As the impulse flows through $\mathcal{T}_r(P)$, it covers an increasing portion of $\mathcal{T}_r(P)$. The point which the impulse reaches at time t on its way from r to some leaf of $\mathcal{T}_r(P)$ is called *corner* at time t. Clearly, there exist at most as many corners as there are leaves in $\mathcal{T}_r(P)$.

By computing all corners at a specific moment in time, and arranging them in the order in which they appear when $\mathcal{T}_r(P)$ is traversed in depth-first manner, it is feasible to construct a closed polygonal chain, a so-called *wavefront* $w(t)$ at time t.

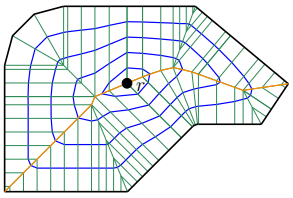
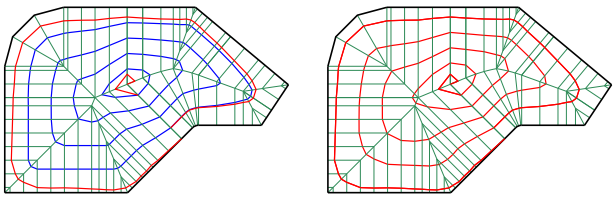


Figure 2: Wavefronts.

In order to guarantee that the user-specified maximum step-over Δ is respected, the spacing of the wavefronts has to be chosen carefully. Roughly, we divide the Euclidean height $h_{\mathcal{T}_r(P)}(r)$ of r by Δ in order to get a lower bound on the number $m+1$ of wavefronts. (In Figure 2, the two longest paths in $\mathcal{T}_r(P)$ between r and leaves of $\mathcal{T}_r(P)$ that correspond to $h_{\mathcal{T}_r(P)}(r)$ are shown in orange.) This gives us a uniform decomposition of time $T := (t_0, t_1, ..., t_m)$, for $m \in \mathbb{N}_0$, with $0 = t_0 < t_1 < ... < t_m = 1$. Then the corners of the wavefront $w(t_i)$ are given by the positions of the impulse at time t_i. (We omit formulas due to lack of space.) Note that this construction implies that the longest paths are split by the wavefronts into sections with length at most Δ. In particular, our construction ensures that the (symmetric) Hausdorff distance between $w(t_i)$ and $w(t_{i+1})$ is at most Δ and, thus, that the maximum step-over Δ is respected. Note that $w(t_m)$ equals ∂P while $w(t_0)$ degenerates to r.

5 One Spiral Path

We now focus on the generation of the actual spiral path, which is fundamentally different to the strategy applied by Held and Spielberger [4]. The spiral path $\mathcal{S}(P, \Delta)$ is made up of m laps $L_1, L_2, \ldots, L_m$. Each of these laps is a polygonal chain whose corners lie on $\mathcal{T}_r(P)$. (In addition, one final (trivial) lap is needed for moving the disk along ∂P.) In a nutshell, we compute the innermost lap L_1 by interpolating between the wavefronts $w(t_0)$, i.e., the root r of $\mathcal{T}_r(P)$, and $w(t_1)$. Similarly, L_m constitutes an interpolation between $w(t_{m-1})$ and $w(t_m)$, i.e., ∂P. All other laps are formed by interpolations between L_1 and L_m, see Figure 3. Every lap starts and ends at one specific clearance line incident at r. The important technical issue is to generate these laps in such a way that the step-over distance between neighboring laps does not exceed the user-specified maximum step-over Δ.



(a) (b)

Figure 3: (a) First and last lap. (b) All laps.

We start with explaining how L_1 is generated, see Figure 4. Recall that $w(t_0)$ degenerates to r. Suppose that q_0 is the vertex of $w(t_1)$ that is intersected by the clearance line $\overline{rv_0}$, on which all laps start and end. Thus, L_1 starts at r and ends at q_0. If we want to generate counter-clockwise (CCW) laps then we number the vertices of $w(t_1)$ in CCW order, starting at q_0. Now consider some vertex of $w(t_1)$, e.g., q_4 in Figure 4. Let d_4 denote the length of the polygonal chain $q_0 q_1 \ldots q_4$, let d denote the circumference of $w(t_1)$, and let δ_4 denote the distance (along $\mathcal{T}_r(P)$) from q_4 to r. Then a candidate corner c of L_1 is placed on the path from q_4 to r at a distance (along $\mathcal{T}_r(P)$) of

$$\left(1 - \frac{d}{d_4}\right) \cdot \delta_4$$

from q_4. We store c at the corresponding edge of $\mathcal{T}_r(P)$. Note that some vertices of $w(t_1)$ might end up storing candidate corners on the same edge or path towards r.

After setting the weight d to the circumference of ∂P and letting the corners of $w(t_{m-1})$ play the role of r, we obtain candidate corners for L_m in a similar way by moving from the vertices of $w(t_m)$, i.e., ∂P, towards the vertices of $w(t_{m-1})$. If required, we can also let L_m wind around r a bit more than once, and let it end at some point on ∂P other than v_0, by making d larger than the circumference of ∂P.

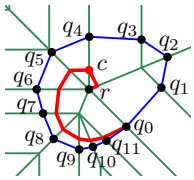


Figure 4: Lap generation.

In order to actually generate L_1 we scan $\mathcal{T}_r(P)$ in a depth-first order, starting at r and moving along $\overline{rv_0}$ as first branch of $\mathcal{T}_r(P)$. The recursive scan stops whenever a candidate corner for L_1 is encountered. This depth-first scan establishes all corners of L_1 in the desired (CCW or CW) order. Then we we start a new depth-first scan towards the leaves of $\mathcal{T}_r(P)$ at every corner q of L_1. The recursion of the depth-first scan from q is stopped whenever we get to a distance $(m-2)\Delta$ from q along $\mathcal{T}_r(P)$. At every such stopping point of the recursion a new candidate corner for L_m is placed. Then another depth-first scan starting at r reveals all corners of L_m by stopping the recursion whenever a candidate corner for L_m is encountered.

We note that this construction guarantees the following distances, where $H(\cdot, \cdot)$ denotes the symmetric Hausdorff distance: $H(r, L_1) \leq \Delta$ and $H(\partial P, L_m) \leq \Delta$ and $H(L_1, L_m) \leq (m-2) \cdot \Delta$.

The remaining laps $L_2, \ldots, L_{m-1}$ can be produced similar to the generation of the initial wavefronts if we take the freedom to regard one lap as a special type of wavefront between L_1 and L_m: Again we let an impulse propagate along $\mathcal{T}_r(P)$. However, this modified impulse propagation starts at time $t = 0$ at the

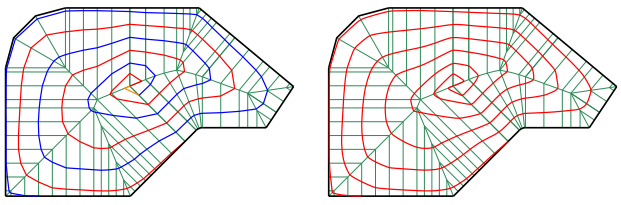
corners of L_1, and ends at time $t = 1$ at the corners of L_m. Then, for properly chosen velocities of the impulse on the edges of $\mathcal{T}_r(P)$, the "wavefront" that corresponds to the time $i/m-2$ forms the lap L_{i+1}, for $i \in \{1, 2, \ldots, m-2\}$.

By connecting all laps in the natural way we obtain a polygonal path $\mathcal{S}(P, \Delta)$ inside P. Trivially, $\mathcal{S}(P, \Delta)$ starts at r and ends on ∂P. Furthermore, $\mathcal{S}(P, \Delta)$ is not self-intersecting, because we are gradually moving outwards, starting at r, until we arrive at ∂P. And due to the construction, $\mathcal{S}(P, \Delta)$ respects the maximum step-over Δ.

6 Generating a Double Spiral

We now generalize our approach to a double spiral that starts and ends at the boundary ∂P. As in the case of a single spiral, the user-specified step-over Δ implies a certain minimum number of wavefronts. For the sake of descriptional simplicity, suppose that this number is odd and that we have $2k + 1$ wavefronts $w(t_0), w(t_1), \ldots, w(t_{2k})$, with $w(t_0)$ equal to r and $w(t_{2k})$ equal to ∂P. We use the algorithm of Section 5 to compute one single spiral with maximum step-over 2Δ which starts at r and ends at v_0 on ∂P. Let $L_1, L_3, \ldots, L_{2k-1}$ denote the successive laps of this spiral. Hence, L_1 starts at r and ends at the intersection q of $w(t_2)$ with $\overline{rv_0}$, L_3 starts at q and ends on $w(t_4)$, and so on. In particular, L_{2k-1} ends at v_0 on ∂P.

Let L_{2k+1} be identical to ∂P. At every corner of lap L_i, for $i \in \{1, 3, \ldots, 2k-1\}$ we plant an impulse that moves towards the leaves of $\mathcal{T}_r(P)$, starting on L_i at time $t = 0$ and reaching L_{i+2} at time $t = 1$. Stopping the impulse at time $t = 1/2$ yields the laps $L_2, L_4, \ldots, L_{2k}$, where L_2 starts at q and L_{2k} ends at v_0 on ∂P. As for a single spiral, the positions of the end-points of L_{2k-1} and L_{2k} on ∂P can be adjusted to meet specific needs. In Figure 5(a), the two sequences of laps are shown in red and blue.



(a) (b)

Figure 5: (a) Double spiral, and (b) its approximation by cubic Bézier curves.

In order to connect the start of L_2 at q with the start of L_1 at r we move from the corners of L_1 towards r for a distance of Δ, thus obtaining corners of a polygonal path that connects L_1 and L_2. In Figure 5(a), this path is shown in orange. This construction ensures that the resulting double spiral is not self-intersecting and respects the maximum step-over Δ.

7 Extensions

As explained in [5], we can decompose a complex (possibly multiply-connected) shape into simpler sub-shapes and then compute spiral paths within these sub-shapes: Two single spirals and several double spirals can be linked to form one continuous multi-spiral path that spirals through the entire shape, see Figure 6. We omit details due to lack of space.

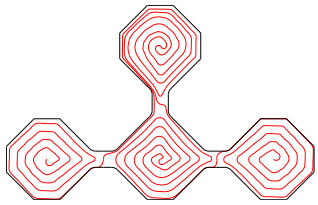


Figure 6: Shape covered by one multi-spiral path.

Furthermore, the polygonal spirals can be approximated by higher-order primitives. For instance, Figure 5(b) shows an approximation by cubic Bézier curves of the double spiral of Figure 5(a). (For the approximation we use the PowerApx package [3].)

References

[1] M. Abrahamsen. Spiral Toolpaths for High-Speed Machining of 2D Pockets With or Without Islands. In *Proc. ASME IDETC/CIE 2015 Conf.*, 2015.

[2] M.B. Bieterman and D.R. Sandstrom. A Curvilinear Tool-Path Method for Pocket Machining. *ASME J. Manufac. Science Eng.*, 125(4):709–715, November 2003.

[3] M. Heimlich and M. Held. Biarc Approximation, Simplification and Smoothing of Polygonal Curves by Means of Voronoi-Based Tolerance Bands. *Internat. J. Comput. Geom. Appl.*, 18(3):221–250, June 2008.

[4] M. Held and C. Spielberger. A Smooth Spiral Tool Path for High Speed Machining of 2D Pockets. *Comput. Aided Design*, 41(7):539–550, July 2009.

[5] M. Held and C. Spielberger. Improved Spiral High-Speed Machining of Multiply-Connected Pockets. *Comput. Aided Design & Appl.*, 11(3):346–357, 2014.

Computing the k-resilience of a Synchronized Multi-Robot System

S. Bereg* L.E. Caraballo† J.M. Díaz-Báñez† M.A. Lopez‡

Abstract

In this work, we introduce the notion of k-resilience in a cooperative system of mobile robots, as a measure of the system's ability to gather information from a set of mobile robots with communication constraints. More formally, the k-resilience is defined as the cardinality of a smallest set of robots whose failure suffices to cause that at least k surviving robots operate without communication. Obviously, the larger the resilience, the more fault tolerant the system is. We prove that the problem of computing the k-resilience is NP-hard when k is part of the input and show that for some specific configurations the problem can be efficiently solved.

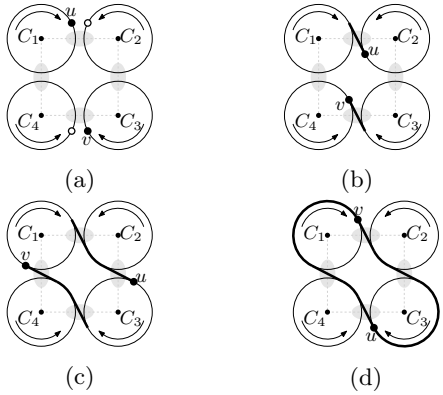


Figure 1: When the robots represented by non-solid points leave the system then the surviving robots, solid points, follow the paths in bold solid stroke.

1 Introduction

A particular type of a synchronized system of mobile autonomous robots to perform tasks cooperatively is presented in [3]. They consider the following simple model that can be generalized to more realistic scenarios: let $T = \{C_1, \ldots, C_n\}$ be a set of n pairwise disjoint unit circles (trajectories) and a team of n robots with a limited communication range ϵ ($0 < \epsilon < 0.5$) traveling along these trajectories with the same constant speed while perform some task in a cooperative way (the communication between the robots is a crucial issue). The *communication graph* $G = (V, E)$ on T is a geometric graph where V is given by the centers of the trajectories in T, and E is formed by the pairs of indexes $\{i, j\}$ such that the distance between the centers of C_i and C_j is at most $2 + \epsilon$. Two circles are *neighboring* if they are adjacent in the communication graph. This work is focused on sets of trajectories whose communication graph is connected.

The *link position* of C_i with respect to C_j, denoted by ϕ_{ij} (the angle of the ray from the center of C_i to the center of C_j), is the point of C_i closest to C_j. Notice that two robots in neighboring circles C_i and C_j can exchange information if they are close enough to the link positions ϕ_{ij} and ϕ_{ji}, respectively. Assume w.l.o.g. that a trajectory can be covered in one time unit. If two robots in neighboring circles C_i and C_j periodically arrive at the same time at ϕ_{ij} and ϕ_{ji} respectively, we say that they are *synchronized*. A system of robots is *synchronized* if every pair of neighbors is synchronized.

If one or more robots leave the system (due to failure, refueling, etc.), then some trajectories are left unattended. To handle such cases in a synchronized system the authors of [3] propose the following *shifting strategy*: when a live robot arrives to a link position and detects the absence of the neighbor, it shifts to the neighboring trajectory in order to assume the unattended task. For synchronization reasons, when a robot enters in a neighboring trajectory C_j, it must follow the direction of travel assigned to C_j. Also, during the shifting process it must accelerate to maintain the schedule. For simplicity, we can assume that shifting is instantaneous and that no time or lenght is spent. Due to the kinematic constraints imposed by real scenarios applying this recovery strategy, the authors of [3] propose to assign opposite movement directions in neighboring circles (one clockwise (CW) and one counterclockwise (CCW)). Consequently, the underlying communication graph must be bipartite.

Figure 1a shows a system where the points on the circles represent the robots gray ovals enclose the link positions between neighboring trajectories. The system is synchronized, and if the robots represented by non-solid points leave the system then the surviving ones, u and v, enter a *starvation* state, permanently failing to meet other robots. To measure the tolerance of a system to this phenomenon it is defined in

*Department of Computer Science, University of Texas at Dallas, USA.
†Department of Applied Mathematics II, University of Seville, SPAIN.
‡Department of Computer Science, University of Denver, USA.

[1] the *k-resilience* as the minimum number of robots whose removal may cause k live robots to starve.

2 Problem statement and properties

Definition 1 *A synchronized communication system* (SCS) *is a set of n robots on a set of n trajectories, one for each trajectory, such that every pair of neighbors are synchronized and move in opposite directions. An* m-partial SCS, $0 < m \leq n$, *is a syncronized system in which $n - m$ robots have left the system and the m remaining robots apply the shifting strategy.*

Note that a SCS is a partial SCS where no robots have left. Thus, any claims about partial SCSs hold for SCSs as well.

Definition 2 (Starvation) *In an m-partial SCS, a robot* starves *or it is in* starvation *if every time that it arrives at a link position the corresponding neighbor is not there causing a shifting to the neighboring trajectory.*

Definition 3 (k-Resilience) *The* k-resilience *of a SCS ($k \geq 1$) is the minimum number of robots whose removal may cause the starvation of at least k surviving robots. If it is not possible to obtain k starving robots then the k-resilience is infinity.*

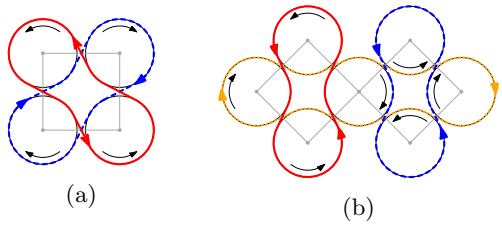
The problem we focus on is: *Given a SCS and a natural number k, determine its k-resilience.*

Definition 4 (Ring) *Let T be a set of trajectories with an assignment of movement directions such that every pair of neighboring circles has opposite directions. A* ring *is the locus of points visited by a robot following the assigned directions and always shifting to the neighboring trajectory in the link positions.*

Note that the movement described by a starving robot is a ring. The trajectories of a SCS can be partitioned into disjoint rings and each ring has a direction of travel determined by the movement direction in the participating trajectories, see Figure 2.

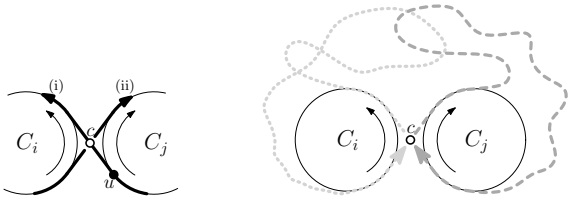
Regarding the length of a ring, it is convenient to ignore the effect on distance arising from the shifts between neighboring trajectories, i.e., to proceed as if neighboring circles are tangent to each other. Therefore, the *length* of a ring is defined as the sum of the lengths of the arcs of trajectories forming it. A *path* in a ring r from a point $a \in r$ to a point $b \in r$ is a sequence of visited points from a to b following the travel direction in r and it may contain tours on r. If a path does not contain any tour in the ring then we say that it is a *single path*.

Lemma 1 *In an m-partial SCS the number of robots in a given ring remains invariant.*



(a) (b)

Figure 2: SCS trajectories are partitioned into rings. (a) an SCS with two rings; (b) three rings.



(a) (b)

Figure 3: (a) Crossing directions. (b) The two ties determined by a point c where a ring crosses itself.

Lemma 2 *In an m-partial SCS, the length of a single path between two robots of a ring is in $2\pi\mathbb{N}$.*

Lemma 3 *In a SCS, the length of every ring is in $2\pi\mathbb{N}$. A ring of length $2l\pi$ has at most l robots in any resultant partial SCS. Furthermore, if no robots have left the system, the ring has exactly l robots, each at distance 2π from the next.*

Lemma 4 *If the communication graph of a set of trajectories is a tree then there is a single ring.*

The following concept gives us a useful tool to study the k-resilience of a SCS.

Definition 5 (Starvation number) *The* starvation number *of a SCS is the maximum possible number of starving robots in a resultant partial SCS.*

Lemma 5 *If the starvation number of a SCS is s then the s-resilience of the system is $n - s$. Furthermore, for all $k > s$ the k-resilience is infinity.*

3 Hardness of Computing the k-Resilience

First, we introduce some notation. The middle point of the edge between two neighboring circles C_i and C_j is called a *crossing point*. A starving robot may pass it in two possible directions, one following the ring from C_i to C_j and other following the ring from C_j to C_i (these two rings could be one and the same), see Figure 3a. We call them *crossing directions*.

Lemma 6 *In an m-partial SCS, let r and r' be rings (not necessarily distinct) that cross each other at a*

point c. Let u and u' be two robots in r and r', respectively. If there are two paths of equal lengths, one from u to c in r (possibly longer than r) and other from u' to c in r' (possibly longer than r'), then, assuming no more failures, u and u' are not starving.

In the situation of the above lemma we say that u' and u *prevent* each other from starving. Notice that:

Observation 1 *In an m-partial SCS, a robot is starving if and only if all the robots that prevent it from starving have failed.*

In an m-partial SCS, if r is a ring that crosses itself at a point c between circles C_i and C_j, then every starving robot in r passes through c periodically and alternating the crossing directions, see Figure 3b.

Definition 6 (Tie of a ring) *A* tie *of a ring is a closed path that starts and ends at a crossing point of the ring with itself (without passing through this crossing point).*

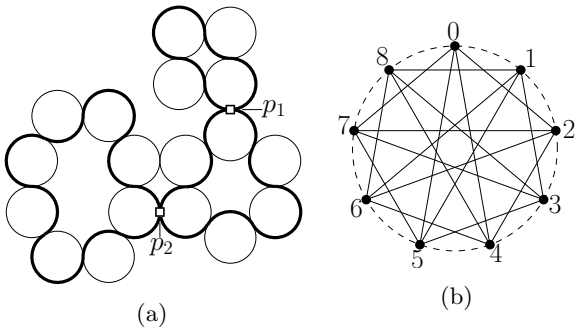
Lemma 7 *In a SCS of n trajectories whose communication graph is a tree, a crossing point determines two ties of lengths $2l\pi$ and $2(n-l)\pi$ respectively, where $l \in \mathbb{N}$ (Figure 3b).*

Theorem 8 *In a m-partial SCS where the m surviving robots are in only one ring, a robot u starves if and only if the length of every single path (in the ring) between u and other live robot is different from the lengths of all ties in the ring.*

Consider an m-partial SCS where the m surviving robots are in a ring r of length $2m\pi$. Let $0, 1, \ldots, m-1$ be an enumeration of the robots in r, following the travel direction of the ring. Lemma 3 implies that robot i is 2π ahead of robot $i-1$ (mod m, as usual). Let $2l_1\pi, \ldots, 2l_t\pi$ be the lengths of all ties in r. By Theorem 8, robot i starves if and only if the robots with indices in $\{i+l_1, \ldots, i+l_t\}$ fail. The relation "prevent from starving" between the robots of a ring can be modeled using an undirected graph whose nodes correspond to the robots in the ring and, for all $i \neq j$, there is an edge between nodes i and j if and only if robots i and j prevent each other from starving. The resulting graph is a *circulant graph*[1]. Figure 4 shows an example. A circulant graph of n nodes can be shortly denoted as C_nS where S is the set of "jumps" between the adjacent vertices.

Lemma 9 *In a SCS, the maximum possible number of starving robots in a ring is equal to the cardinality of the maximum* independent set *in the corresponding circulant graph.*

[1]Given $n \in \mathbb{N}$ $(n > 1)$ and $S \subseteq \{1, 2, \ldots, n-1\}$, the *circulant graph* of n nodes and set of *jumps* S is the graph whose sets of nodes and edges are $\{0, 1, \ldots, n-1\}$ and $\{(i, (i+e) \mod n) \mid \forall i \in [1, n], \forall e \in S\}$, respectively.



(a) (b)

Figure 4: (a) Example of a SCS. With bold solid stroke a ring that crosses itself twice at p_1 and p_2. (b) The circulant graph that models the relation "prevent from starving" corresponding to the ring in (a).

In the following we define an auxiliary operation to transform a circulant graph into another circulant graph with some interesting properties.

Definition 7 ($K_{n,n}$-augmentation) *Let $G = (V, E)$ be a graph with n nodes. A graph $G' = (V', E')$ is a* clone *of G if G' and G are isomorphic and $V \cap V' = \emptyset$. The $K_{n,n}$-*augmentation *of G, denoted by $\overline{G} = (\overline{V}, \overline{E})$, is the result of the graph join operation between G and G', i.e $\overline{V} = V \cup V'$ and $\overline{E} = E \cup E' \cup \{\{v, w\} \mid v \in V, w \in V'\}$.*

Lemma 10 *Let $G = (V, E)$ and $\overline{G} = (\overline{V}, \overline{E})$ be a graph and its $K_{n,n}$-augmentation, respectively. G is a circulant graph if and only if $\overline{G}$ is a circulant graph. Moreover, the maximum independent set of G and the maximum independent set of $\overline{G}$ have the same cardinality.*

Theorem 11 *The problem of computing the starvation number of a SCS (SN-SCS) is NP-hard, even, if the communication graph is a caterpillar tree*[2]*.*

Proof. (Sketch) We use a reduction from the problem of computing the Maximum Independent Set in a Circulant Graph (MIS-CG) which is NP-hard [2]. Let C_nS be a circulant graph with $n \geq 2$, as input for the MIS-CG problem. Finding the maximum independent set in C_nS is equivalent to obtaining the maximum independent set in its $K_{n,n}$-augmentation. Then, for convenience we work with $C_{2n}\overline{S}$ (Lemma 10) which is the $K_{n,n}$-augmentation of the given circulant graph C_nS. By Lemma 9, it suffices to transform $C_{2n}\overline{S}$ into a SCS of $2n$ circles whose communication graph is a caterpillar tree such that:

$$d \in \overline{S} \Leftrightarrow \text{there is a tie of length } 2d\pi \text{ in the SCS} \quad (1)$$

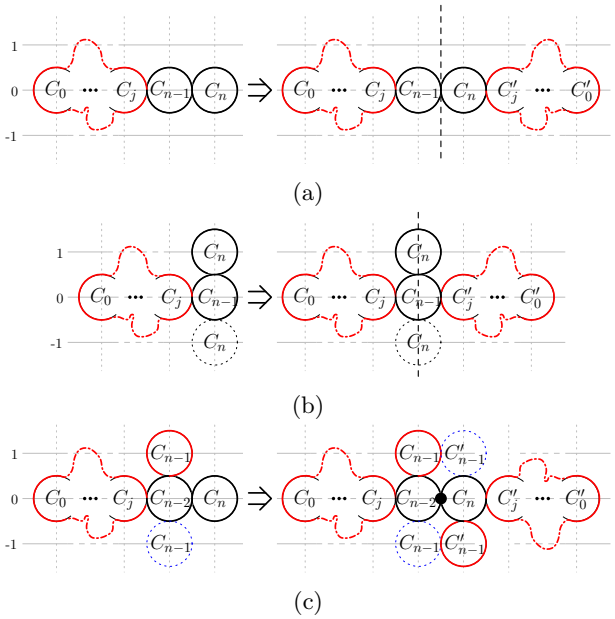


Figure 5: (a) If C_{n-1} and C_n are both on the 0-line, then apply symmetry about the vertical line between C_{n-1} and C_n. (b) If C_{n-1} is on the 0-line but C_n is not, then apply symmetry about the vertical line passing through the center of C_{n-1}. (c) Otherwise, apply symmetry about the touching point of C_{n-2} and C_n.

We place the circles on three horizontal lines with coordinates in 1, 0 and -1. First, place the circle C_0 on the 0-line. Then place $C_i, i = 1, \ldots, n$ as follows. Let C_j be the last circle placed on the 0-line. If $i \in \overline{S}$ then add the circle C_i on 0-line touching C_j. If $i \notin \overline{S}$ then add the circle C_i touching C_j and centered at the 1-line or -1-line in alternate manner, i.e, if the last added circle not centered at the 0-line is centered at the 1-line then add C_i centered at the -1-line, and vice-versa. Notice that i in the second case is even since $\overline{S}$ contains all odd numbers in $[1, n]$. Thus, the next circle C_{i+1} will be placed on the 0-line. Since the lines 1 and -1 are alternating, C_i touches only one circle C_j. We have placed $n+1$ circles $C_0, \ldots, C_n$. In order to add the $n-1$ remaining circles we proceed as follow:

- If n is even: if $n \in \overline{S}$ we proceed as shown in Figure 5a; if $n \notin \overline{S}$ then we proceed as shown in Figure 5b.
- If n is odd: if $(n-1) \in \overline{S}$ then we proceed as shown in Figure 5a; if $(n-1) \notin \overline{S}$ then we proceed as shown in Figure 5c.

Now, statement (1) can be proven on the obtained SCS and this completes the proof.

□

[2] A caterpillar tree is a tree in which all the vertices are within distance 1 of a central path.

The main result of this section then follows from Theorem 11 and Lemma 5:

Theorem 12 *The problem of computing the k-resilience of a SCS is NP-hard.*

4 The k-resilience for cycles and grid-graphs

Lemma 13 *Let G be the communication graph of an m-partial SCS. If G is a cycle, then the system has two rings, with different travel directions. Furthermore, every edge of G corresponds to a crossing of the two rings. (See Figure 2(a)).*

Lemma 14 *In an m-partial SCS, whose communication graph is a cycle with rings r and r', a robot in r is starving if and only if r' is empty of robots.*

Theorem 15 *Consider a system whose communication graph is a cycle and let r and r' be two rings with lengths $2\pi l$ and $2\pi l'$, respectively. The starvation number of the system is $\max\{l, l'\}$ and the k-resilience is $\min\{l, l'\}$ if $k \leq \max\{l, l'\}$ and* infinity, *otherwise.*

Theorem 16 *If the communication graph G of an m-partial SCS is a grid then, two starving robots can not occupy trajectories in the same row (resp. column).*

Theorem 16 implies that in a grid-graph there is at most one starving robot per column or row. Consequently, using induction on k and the pigeonhole principle, the following result can be deduced:

Theorem 17 *The starvation number in an $n \times m$ grid system is $\min(n, m)$ and its k-resilience is $k(n+m-2) - k(k-1)$ if $k \leq \min(n, m)$ and* infinity, *otherwise.*

Acknowledgments

This work has been partially supported by the projects MTM2016-76272-R (AEI/FEDER,UE) and CONNECT2016-734922 (EU-H2020/MSCA).

References

[1] A. P. Brunner. Isolation in synchronized drone formations, 2015. Master Thesis, University of Denver.

[2] B. Codenotti, I. Gerace, and S. Vigna. Hardness results and spectral techniques for combinatorial problems on circulant graphs. *Linear Algebra and its Applications*, 285(1):123 – 142, 1998.

[3] J.M. Díaz-Báñez, L.E. Caraballo, M.A. López, S. Bereg, I. Maza, and A. Ollero. The synchronization problem for information exchange between aerial robots under communication constraints. In *International Conference on Robotics and Automation (ICRA)*. IEEE, 2015.

A superlinear lower bound on the number of 5-holes*

Oswin Aichholzer† Martin Balko‡§ Thomas Hackl† Jan Kynčl‡ Irene Parada†
Manfred Scheucher†§ Pavel Valtr‡§ Birgit Vogtenhuber†

Abstract

Let P be a finite set of points in the plane in *general position*, that is, no three points of P are on a common line. We say that a set H of five points from P is a 5-*hole in* P if H is the vertex set of a convex 5-gon containing no other points of P. For a positive integer n, let $h_5(n)$ be the minimum number of 5-holes among all sets of n points in the plane in general position.

Despite many efforts in the last 30 years, the best known asymptotic lower and upper bounds for $h_5(n)$ have been of order $\Omega(n)$ and $O(n^2)$, respectively. We show that $h_5(n) = \Omega(n \log^{4/5} n)$, obtaining the first superlinear lower bound on $h_5(n)$.

The following structural result, which might be of independent interest, is a crucial step in the proof of this lower bound. If a finite set P of points in the plane in general position is partitioned by a line ℓ into two subsets, each of size at least 5 and not in convex position, then ℓ intersects the convex hull of some 5-hole in P. The proof of this result is computer-assisted.

1 Introduction

We say that a set of points in the plane is in *general position* if it contains no three points on a common line. A point set is in *convex position* if it is the vertex set of a convex polygon. Let P be a finite set of points in general position in the plane. We say that a set H of k points from P is a k-*hole in* P if H is the vertex set of a convex polygon containing no other points of P.

In the 1970s, Erdős [6] asked whether, for every positive integer k, there is a k-hole in every sufficiently large finite point set in general position in the plane. Harborth [8] proved that there is a 5-hole in every set of 10 points in general position in the plane and gave a construction of 9 points in general position with no 5-hole. After unsuccessful attempts of researchers to answer Erdős' question affirmatively for any fixed integer $k \geq 6$, Horton [9] constructed, for every positive integer n, a set of n points in general position in the plane with no 7-hole. The question whether there is a 6-hole in every sufficiently large finite planar point set remained open until 2007 when Gerken [7] and Nicolás [10] independently gave an affirmative answer.

For positive integers n and k, let $h_k(n)$ be the minimum number of k-holes in a set of n points in general position in the plane. Due to Horton's construction [9], $h_k(n) = 0$ for every n and every $k \geq 7$. The functions $h_3(n)$ and $h_4(n)$ are both known to be asymptotically quadratic [2, 4]. For $h_5(n)$ and $h_6(n)$, the best known asympotic bounds are $\Omega(n)$ and $O(n^2)$ [4, 7, 8, 10]. See, e.g., [2] for more details.

As our main result, we give the first superlinear lower bound on $h_5(n)$. This solves an open problem, which was explicitly stated, for example, in a book by Brass, Moser, and Pach [5, Chapter 8.4, Problem 5] and in the survey [1].

Theorem 1 *There is a fixed constant $c > 0$ such that for every integer $n \geq 10$ we have $h_5(n) \geq cn \log^{4/5} n$.*

Let P be a finite set of points in the plane in general position and let ℓ be a line that contains no point of P and that partitions P into two non-empty subsets A and B. We then say that $P = A \cup B$ is ℓ-*divided*.

The following result, which might be of independent interest, is a crucial step in the proof of Theorem 1.

Theorem 2 *Let $P = A \cup B$ be an ℓ-divided set with $|A|, |B| \geq 5$ and with neither A nor B in convex position. Then there is an ℓ-divided 5-hole in P.*

*The research for this article was partially carried out in the course of the bilateral research project "Erdős–Szekeres type questions for point sets" between Graz and Prague, supported by the OEAD project CZ 18/2015 and project no. 7AMB15A T023 of the Ministry of Education of the Czech Republic. Aichholzer, Scheucher, and Vogtenhuber were partially supported by the ESF EUROCORES programme EuroGIGA – CRP ComPoSe, Austrian Science Fund (FWF): I648-N18. Parada was supported by the Austrian Science Fund (FWF): W1230. Balko and Valtr were partially supported by the grant GAUK 690214. Balko, Kynčl, and Valtr were partially supported by the project CE-ITI no. P202/12/G061 of the Czech Science Foundation (GAČR). Hackl and Scheucher were partially supported by the Austrian Science Fund (FWF): P23629-N18. Balko, Scheucher, and Valtr were partially supported by the ERC Advanced Research Grant no 267165 (DISCONV).

†Institute for Software Technology, Graz University of Technology, Austria, [oaich,iparada,mscheuch,bvogt]@ist.tugraz.at

‡Department of Applied Mathematics and Institute for Theoretical Computer Science, Faculty of Mathematics and Physics, Charles University, Czech Republic, [balko,kyncl]@kam.mff.cuni.cz

§Alfréd Rényi Institute of Mathematics, Hungarian Academy of Sciences, Budapest, Hungary

The proof of Theorem 2 is computer-assisted. We reduce the result to several statements about point sets of size at most 11 and then verify each of these statements by an exhaustive computer search. To verify the computer-aided proofs we have implemented two independent programs, which, in addition, are based on different abstractions of point sets. Some of the tools that we use originate from a bachelor's thesis of Scheucher [12].

In the rest of the paper, we assume that every point set P is planar, finite, and in general position. We also assume, without loss of generality, that all points in P have distinct x-coordinates. We use $\mathrm{conv}(P)$ to denote the convex hull of P and $\partial\,\mathrm{conv}(P)$ to denote the boundary of the convex hull of P.

A subset Q of P that satisfies $P \cap \mathrm{conv}(Q) = Q$ is called an *island of* P. Note that every k-hole in an island of P is also a k-hole in P.

2 Proof of Theorem 1

We now show how to apply Theorem 2 to obtain a superlinear lower bound on the number of 5-holes in a given set of n points. Without loss of generality, we assume that $n = 2^t$ for some integer $t \geq 5^5$.

We prove by induction on $t \geq 5^5$ that the number of 5-holes in an arbitrary set P of $n = 2^t$ points is at least $f(t) := c \cdot 2^t t^{4/5} = c \cdot n \log_2^{4/5} n$ for some absolute constant $c > 0$. For $t = 5^5$, we have $n > 10$ and, by the result of Harborth [8], there is at least one 5-hole in P. If c is sufficiently small, then $f(t) = c \cdot n \log_2^{4/5} n \leq 1$ and we have at least $f(t)$ 5-holes in P, which constitutes our base case.

For the inductive step we assume that $t > 5^5$. We first partition P with a line ℓ into two sets A and B of size $n/2$ each. Then we further partition A and B into smaller sets using the following well-known lemma, which is, for example, implied by a result of Steiger and Zhao [13, Theorem 1].

Lemma 3 ([13]) *Let $P' = A' \cup B'$ be an ℓ-divided set and let r be a positive integer such that $r \leq |A'|, |B'|$. Then there is a line disjoint from P' that determines an open halfplane h with $|A' \cap h| = r = |B' \cap h|$.*

We set $r := \lfloor \log_2^{1/5} n \rfloor$, $s := \lfloor n/(2r) \rfloor$, and apply Lemma 3 iteratively in the following way to partition P into islands $P_1, \ldots, P_{s+1}$ of P so that the sizes of $P_i \cap A$ and $P_i \cap B$ are exactly r for every $i \in \{1, \ldots, s\}$. Let $P'_0 := P$. For every $i = 1, \ldots, s$, we consider a line that is disjoint from P'_{i-1} and that determines an open halfplane h with $|P'_{i-1} \cap A \cap h| = r = |P'_{i-1} \cap B \cap h|$. Such a line exists by Lemma 3 applied to the ℓ-divided set P'_{i-1}. We then set $P_i := P'_{i-1} \cap h$, $P'_i := P'_{i-1} \setminus P_i$, and continue with $i + 1$. Finally, we set $P_{s+1} := P'_s$.

For every $i \in \{1, \ldots, s\}$, if one of the sets $P_i \cap A$ and $P_i \cap B$ is in convex position, then there are at least $\binom{r}{5}$ 5-holes in P_i and, since P_i is an island of P, we have at least $\binom{r}{5}$ 5-holes in P. If this is the case for at least $s/2$ islands P_i, then, given that $s = \lfloor n/(2r) \rfloor$ and thus $s/2 \geq \lfloor n/(4r) \rfloor$, we obtain at least $\lfloor n/(4r) \rfloor \binom{r}{5} \geq c \cdot n \log_2^{4/5} n$ 5-holes in P for a sufficiently small $c > 0$.

We thus further assume that for more than $s/2$ islands P_i, neither of the sets $P_i \cap A$ nor $P_i \cap B$ is in convex position. Since $r = \lfloor \log_2^{1/5} n \rfloor \geq 5$, Theorem 2 implies that there is an ℓ-divided 5-hole in each such P_i. Thus there is an ℓ-divided 5-hole in P_i for more than $s/2$ islands P_i. Since each P_i is an island of P and since $s = \lfloor n/(2r) \rfloor$, we have more than $s/2 \geq \lfloor n/(4r) \rfloor$ ℓ-divided 5-holes in P. As $|A| = |B| = n/2 = 2^{t-1}$, there are at least $f(t-1)$ 5-holes in A and at least $f(t-1)$ 5-holes in B by the inductive assumption. Since A and B are separated by ℓ, we have at least

$$\begin{aligned} 2f(t-1) + n/(4r) &= 2c(n/2)\log_2^{4/5}(n/2) + n/(4r) \\ &\geq cn(t-1)^{4/5} + n/(4t^{1/5}) \end{aligned}$$

5-holes in P. The right side of the above expression is at least $f(t) = cnt^{4/5}$, because the inequality $cn(t-1)^{4/5} + n/(4t^{1/5}) \geq cnt^{4/5}$ is equivalent to the inequality $(t-1)^{4/5}t^{1/5} + 1/(4c) \geq t$, which is true if c is sufficiently small, as $(t-1)^{4/5}t^{1/5} \geq t - 1$. This completes the proof of Theorem 1.

3 Preliminaries

Before proceeding with the proof of Theorem 2, we first introduce some notation and definitions, and state some immediate observations.

Let a, b be two points in the plane. We denote the ray starting at a and going through b as $\overrightarrow{ab}$ and the line through a and b directed from a to b as $\overline{ab}$.

Let $P = A \cup B$ be an ℓ-divided set. In the rest of the paper, we assume without loss of generality that ℓ is vertical and directed upwards, A is to the left of ℓ, and B is to the right of ℓ.

ℓ-critical sets An ℓ-divided set $C = A \cup B$ is *ℓ-critical* if it fulfills the following two conditions.

(i) Neither A nor B is in convex position.

(ii) For every extremal point x of C, either $(C \setminus \{x\}) \cap A$ or $(C \setminus \{x\}) \cap B$ is in convex position.

a-wedges and a^*-wedges Let $P = A \cup B$ be an ℓ-divided set. For a point a in A, the rays $\overrightarrow{aa'}$ for all $a' \in A \setminus \{a\}$ partition the plane into $|A| - 1$ regions. We call the closures of those regions *a-wedges* and label them as $W_1^{(a)}, \ldots, W_{|A|-1}^{(a)}$ in clockwise order around a, where $W_1^{(a)}$ is the topmost a-wedge that intersects ℓ. Let $t^{(a)}$ be the number of a-wedges that intersect ℓ. Note that $W_1^{(a)}, \ldots, W_{t^{(a)}}^{(a)}$ are the a-wedges that intersect ℓ sorted in top-to-bottom order on ℓ. Also note

that all a-wedges are convex if a is an inner point of A, and that there exists exactly one non-convex a-wedge otherwise.

If A is not in convex position, we denote the rightmost inner point of A as a^* and write $t := t^{(a^*)}$ and $W_k := W_k^{(a^*)}$ for $k = 1, \ldots, |A| - 1$. Recall that a^* is unique, since all points have distinct x-coordinates. We set $w_k := |B \cap W_k|$ and label the points of A so that W_k is bounded by the rays $\overrightarrow{a^* a_{k-1}}$ and $\overrightarrow{a^* a_k}$ for $k = 1, \ldots, |A| - 1$. Figure 1 gives an illustration.

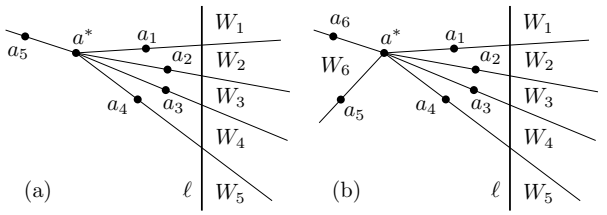


Figure 1: (a) An example of a^*-wedges with $t = |A| - 1$. (b) An example of a^*-wedges with $t < |A| - 1$.

4 Proof of Theorem 2

In the rest of the paper, we state results that we use to prove Theorem 2 and we then present the proof of this theorem. Due to lack of space, we omit the proofs of almost all auxiliary results.

4.1 a^*-wedges with at most two points of B

We first consider an ℓ-divided set $P = A \cup B$ with A not in convex position. We show that, if there is a sequence of consecutive a^*-wedges where the first and the last a^*-wedge both contain two points of B and every a^*-wedge in between them contains exactly one point of B, then there is an ℓ-divided 5-hole in P.

Lemma 4 *Let $P = A \cup B$ be an ℓ-divided set with A not in convex position and with $|A| \geq 5$ and $|B| \geq 6$. Let $W_i, \ldots, W_j$ be consecutive a^*-wedges with $1 \leq i < j \leq t$, $w_i = 2 = w_j$, and $w_k = 1$ for every k with $i < k < j$. Then there is an ℓ-divided 5-hole in P.*

The proof of this lemma is carried out by a rather elaborate case distinction, which we omit here.

4.2 Computer-assisted results

We now provide lemmas that are key ingredients in the proof of Theorem 2. All these lemmas have computer-aided proofs. Each result was verified by two independent implementations, which are also based on different abstractions of point sets. In particular, to prove these lemmas, we employ an exhaustive computer search through all combinatorially different sets of $|P| \leq 11$ points in the plane. Both programs and detailed information are available online [3, 11].

Lemma 5 *Let $P = A \cup B$ be an ℓ-divided set with $|A| = 5$, $|B| = 6$, and with A not in convex position. Then there is an ℓ-divided 5-hole in P.*

Lemma 6 *Let $P = A \cup B$ be an ℓ-divided set with no ℓ-divided 5-hole in P, $|A| = 5$, $4 \leq |B| \leq 6$, and with A in convex position. Then for every point a of A, every convex a-wedge contains at most two points of B.*

Lemma 7 *Let $P = A \cup B$ be an ℓ-divided set with no ℓ-divided 5-hole in P, $|A| = 6$, and $|B| = 5$. Then for each point a of A, every convex a-wedge contains at most two points of B.*

Lemma 8 *Let $P = A \cup B$ be an ℓ-divided set with no ℓ-divided 5-hole in P, $5 \leq |A| \leq 6$, $|B| = 4$, and with A in convex position. Then for every point a of A, if the non-convex a-wedge contains no point of B, every a-wedge contains at most two points of B.*

4.3 Applications of the computer-assisted results

As a first application of the computer-assisted results we prove the following statement, which restricts the number of points of B in a^*-wedges. Its proof uses Lemmas 6, 7, and 8 and also Lemma 4.

Lemma 9 *Let $P = A \cup B$ be an ℓ-divided set with no ℓ-divided 5-hole in P, with $|A| \geq 6$, and with A not in convex position. Then the following two conditions are satisfied.*

(i) Let W_i, W_{i+1}, W_{i+2} be three consecutive a^*-wedges whose union is convex and contains at least four points of B. Then $w_i, w_{i+1}, w_{i+2} \leq 2$.

(ii) Let $W_i, W_{i+1}, W_{i+2}, W_{i+3}$ be four consecutive a^*-wedges whose union is convex and contains at least four points of B. Then $w_i, w_{i+1}, w_{i+2}, w_{i+3} \leq 2$.

4.4 Extremal points of ℓ-critical sets

The following statement, whose relatively easy proof is omitted in this abstract, says that every ℓ-critical set has at most two extremal points on each side of ℓ.

Lemma 10 *Let $C = A \cup B$ be an ℓ-critical set with $|A \cap C| \geq 5$. Then $|A \cap \partial \operatorname{conv}(C)| \leq 2$. By symmetry, an analogous statement holds for B.*

Now we use Lemma 9 to restrict the parameters w_i. Then we state the last auxiliary result used in the proof of Theorem 2.

Lemma 11 *Let $C = A \cup B$ be an ℓ-critical set with no ℓ-divided 5-hole in C and with $|A| \geq 6$. Then $w_i \leq 3$ for every $1 < i < t$. Moreover, if $|A \cap \partial \operatorname{conv}(C)| = 2$, then also $w_1, w_t \leq 3$.*

Proof. Recall that, since C is ℓ-critical, we have $|B| \geq 4$. Let i be an integer with $1 \leq i \leq t$. We assume that there is a point a in $A \cap \partial \operatorname{conv}(C)$, which lies outside of W_i, as otherwise there is nothing to prove for W_i (either $|A \cap \partial \operatorname{conv}(C)| = 1$ and $i \in \{1, t\}$ or $|A \cap \partial \operatorname{conv}(C)| = 2$ and $W_i \cap B = \emptyset$). We consider $C' := C \setminus \{a\}$. Since C is an ℓ-critical set, $A' := C' \cap A$ is in convex position. Thus, there is a non-convex a^*-wedge W' of C'. Since W' is non-convex, all other a^*-wedges of C' are convex. Moreover, since W' is the union of the two a^*-wedges of C that contain a, all other a^*-wedges of C' are also a^*-wedges of C. Let W be the union of all a^*-wedges of C that are not contained in W'. Note that W is convex and contains at least $|A| - 3 \geq 3$ a^*-wedges of C. Since $|A| \geq 6$, the lemma follows from Lemma 9(i). □

Proposition 12 *Let $C = A \cup B$ be an ℓ-critical set with no ℓ-divided 5-hole in C and with $|A|, |B| \geq 6$. Then the following two conditions are satisfied.*

(i) If $|A \cap \partial \operatorname{conv}(C)| = 2$ then $|B| \leq |A| - 1$.

(ii) If $|B \cap \partial \operatorname{conv}(C)| = 2$ then $|B| \leq |A|$.

In the proof of this statement, we use the restrictions from Lemmas 4 and 9 that bound the number of points of B in a^*-wedges to derive the desired bound $|B| \leq |A|$. In the proof of part (i) we also apply Lemma 11 to show strict inequality. The proof of Proposition 12 is quite involved and we omit it here.

4.5 Finalizing the proof of Theorem 2

We are now ready to prove Theorem 2. Namely, we show that for every ℓ-divided set $P = A \cup B$ with $|A|, |B| \geq 5$ and with neither A nor B in convex position there is an ℓ-divided 5-hole in P.

Suppose for the sake of contradiction that there is no ℓ-divided 5-hole in P. We know by the result of Harborth [8] that every set P of ten points contains a 5-hole in P. In the case $|A|, |B| = 5$, the statement then follows from the assumption that neither of A and B is in convex position.

So assume that at least one of the sets A and B has at least six points. We obtain an island Q of P by iteratively removing extremal points so that neither part is in convex position after the removal and until one of the following conditions holds.

(i) One of the parts $Q \cap A$ and $Q \cap B$ has five points.

(ii) Q is an ℓ-critical island of P with $|Q \cap A| \geq 6$ and $|Q \cap B| \geq 6$.

In case (i), we have $|Q \cap A| = 5$ or $|Q \cap B| = 5$. If $|Q \cap A| = 5$ and $|Q \cap B| \geq 6$, then we let Q' be the union of $Q \cap A$ with the six leftmost points of $Q \cap B$. Since $Q \cap A$ is not in convex position, Lemma 5 implies that there is an ℓ-divided 5-hole in Q', which is also an ℓ-divided 5-hole in Q, since Q' is an island of Q. However, this is impossible as then there is an ℓ-divided 5-hole in P because Q is an island of P. We proceed analogously if $|Q \cap A| \geq 6$ and $|Q \cap B| = 5$.

In case (ii), we have $|Q \cap A|, |Q \cap B| \geq 6$. There is no ℓ-divided 5-hole in Q, since Q is an island of P. By Lemma 10, we can assume without loss of generality that $|A \cap \partial \operatorname{conv}(Q)| = 2$. Then it follows from Proposition 12(i) that $|Q \cap B| < |Q \cap A|$. By exchanging the roles of $Q \cap A$ and $Q \cap B$ and by applying Proposition 12(ii), we obtain that $|Q \cap A| \leq |Q \cap B|$, a contradiction. This finishes the proof of Theorem 2.

References

[1] O. Aichholzer. [Empty] [colored] k-gons. Recent results on some Erdős–Szekeres type problems. In *Proceedings of XIII Encuentros de Geometría Computacional*, pages 43–52, 2009.

[2] O. Aichholzer, R. Fabila-Monroy, T. Hackl, C. Huemer, A. Pilz, and B. Vogtenhuber. Lower bounds for the number of small convex k-holes. *Comput. Geom.*, 47(5):605–613, 2014.

[3] M. Balko. http://kam.mff.cuni.cz/˜balko/superlinear5Holes.

[4] I. Bárány and P. Valtr. Planar point sets with a small number of empty convex polygons. *Studia Sci. Math. Hungar.*, 41(2):243–266, 2004.

[5] P. Brass, W. Moser, and J. Pach. *Research Problems in Discrete Geometry.* Springer, 2005.

[6] P. Erdős. Some more problems on elementary geometry. *Austral. Math. Soc. Gaz.*, 5(2):52–54, 1978.

[7] T. Gerken. Empty convex hexagons in planar point sets. *Discrete Comput. Geom.*, 39(1–3):239–272, 2008.

[8] H. Harborth. Konvexe Fünfecke in ebenen Punktmengen. *Elem. Math.*, 33:116–118, 1978.

[9] J.D. Horton. Sets with no empty convex 7-gons. *Canad. Math. Bull.*, 26(4):482–484, 1983.

[10] C.M. Nicolás. The empty hexagon theorem. *Discrete Comput. Geom.*, 38(2):389–397, 2007.

[11] M. Scheucher. http://www.ist.tugraz.at/scheucher/5holes.

[12] M. Scheucher. Counting convex 5-holes, Bachelor's thesis, 2013.

[13] W. Steiger and J. Zhao. Generalized ham-sandwich cuts. *Discrete Comput. Geom.*, 44(3):535–545, 2010.

Classification of empty lattice 4-simplices

Óscar Iglesias-Valiño and Francisco Santos*
(oscar.iglesias@unican.es, francisco.santos@unican.es)
Department of Mathematics, Statistics and Computer Science,
University of Cantabria, SPAIN

Abstract

Combining an upper bound on the volume of empty lattice 4-simplices of large width with a computer enumeration we prove the following conjecture of Haase and Ziegler (2000): *Except for* 179 *classes, of determinant at most* 179*, all empty* 4*-simplices have width one or two with respect to some integer functional.*

1 Motivation and statement of result

A *lattice polytope* is the convex hull of a finite set of integer points (a. k. a. *lattice points*) in $\mathbb{R}^d$. Its *dimension* is the dimension of its affine span. If its vertices are affinely independent it is a *lattice simplex*. A lattice polytope P is called *hollow* if all lattice points of P lie in the boundary of it, and *empty* if all lattice points of P are vertices of it. Equivalently, an empty d-simplex is a lattice polytope whose only lattice points are $d+1$ affinely independent points.

Empty simplices are the fundamental building blocks in the theory of lattice polytopes, in the sense that every lattice polytope P can be triangulated into empty simplices. (Consider, for example, a Delaunay triangulation of the set of lattice points in P). In particular, it is very useful to have classifications or, at least, structural results, concerning the list of all empty simplices in a given dimension. The natural classification of lattice polytopes is modulo *unimodular equivalence*: two lattice polytopes P and Q are $\mathbb{Z}$*-equivalent* or *unimodularly equivalent* (in symbols, $P \cong_{\mathbb{Z}} Q$) if there is an affine transformation $f : \mathbb{R}^n \to \mathbb{R}^n$ with $f(\mathbb{Z}^n) = \mathbb{Z}^n$ and $f(P) = Q$. The name "unimodular" comes from the fact that such a transformation necessarily has determinant ± 1.

Classification of empty 2-simplices is trivial as a consequence of Pick's Theorem [8]: Every empty triangle is unimodularly equivalent to the standard *unimodular triangle* $\mathrm{conv}\{(0,0),(1,0),(0,1)\}$. Here, an empty simplex is called *unimodular* if its *determinant* equals ± 1, where the determinant of $\mathrm{conv}(v_0,\dots,v_d) \subset \mathbb{R}^d$ is defined as

$$\begin{vmatrix} v_0 & \dots & v_d \\ 1 & \dots & 1 \end{vmatrix}.$$

In higher dimensions all unimodular simplices are equivalent, and in particular empty, but they are no longer the only empty simplices. Still, a quite simple classification due to White exists in dimension 3:

Theorem 1 ([11], see also, e.g., [10]) *Every empty tetrahedron of determinant q is equivalent to*

$$T(p,q) := \mathrm{conv}\{(0,0,0),(1,0,0),(0,0,1),(p,q,1)\}$$

for some $p \in \mathbb{Z}$ with $\gcd(p,q) = 1$. Moreover, $T(p,q) \cong_{\mathbb{Z}} T(p',q)$ if and only if $p' = \pm p^{\pm 1} \pmod q$.

A key step in the proof of this theorem is the fact that all empty 3-simplices have *lattice width* equal to one. Here we call *width* of a body $K \subset \mathbb{R}^d$ with respect to a linear functional $f : \mathbb{R}^d \to \mathbb{R}$ the difference $\max_{x\in K} f(x) - \min_{x\in K} f(x)$. We call *(lattice) width* of a lattice polytope P the minimum width of P with respect to integer functionals. For example, the empty 3-simplices used in Theorem 1 have width one with respect to the functional $f(x,y,z) = z$.

As an illustration of how useful classifying empty simplices is, let us mention the following result of Kantor and Sarkaria [6] improved by Santos and Ziegler [9]. For a lattice polytope P and a $c \in \mathbb{N}$ we denote by cP the dilation of P by a factor c. We say that a polytope P has a *unimodular triangulation* if it admits a triangulation into unimodular lattice simplices. We say it has a *unimodular cover* if it can be covered by unimodular simplices contained in P.

Theorem 2 ([6, 9]) *Let P be a lattice 3-polytope:*

1. *$2P$ has a unimodular cover, but not necessarily a unimodular triangulation.*
2. *cP has a unimodular triangulation for all $c \in \{4\} \cup \{6,7,8,9,\dots\}$, .*

Proof. (Sketch). Since P can by triangulated into empty 3-simplices, in part (1) there is no loss of generality in assuming that P is an empty simplex. The

*Supported by grants MTM2014-54207-P (both authors) and BES-2015-073128 (O. Iglesias) of the Spanish Ministry of Economy and Competitiveness. F. Santos is also supported by the Einstein Foundation Berlin. This work has been presented as a poster in the *Einstein Worksohp on Lattice Polytopes* and the *Jornada de Topología y Combinatoria*.

classification of empty 3-simplices easily gives the result. Part (2) is a bit more complicated because one needs to make sure that the dilated empty 3-simplices that triangulate cP admit unimodular triangulations that are compatible in common boundaries. Still, thanks to the classification, this is doable. □

In dimension 4 a full classification of empty 4-simplices is not known, but the following facts are:

1. There are infinitely many emtpy 4-simplices of width one (e.g., cones over empty tetrahedra).
2. There are infinitely many emtpy 4-simplices of width two (Haase and Ziegler [5]).
3. There are *finitely many* empty 4-simplices of width greater than two (Blanco et al. [3]).
4. Every empty 4-simplex P is *cyclic*, meaning that $\mathbb{Z}^4/\Lambda(P)$ is a cyclic group [2]. Here, $\Lambda(P) \subset \mathbb{Z}^4$ denotes the sublattice generated by the edge-vectors of P. (Observe that for every lattice simplex of determinant D, $\mathbb{Z}^d/\Lambda(P)$ is a finite abelian group of order D).

Via an exhaustive computer enumeration, Haase and Ziegler [5] also proved:

Theorem 3 ([5]) *Among the empty 4-simplices of determinant $D \leq 1000$,*

1. *There are no simplices of width ≥ 5.*
2. *There is a unique equivalence class of simplices of width 4. This class has determinant $D = 101$.*
3. *There are exactly 178 classes of width 3, with determinants between 41 and 179.*

Based on this, they conjectured that all empty 4-simplices of determinant beyond 179 have width one or two. The main result in this work is a proof of this:

Theorem 4 *All empty 4-simplices of width greater than two have determinant at most 179. Hence, there are exactly 179 classes of them, as computed by Haase and Ziegler [5]).*

We prove Theorem 4 combining a theoretical upper bound for the determinant of lattice simplices of large width and a computer enumeration of empty 4-simplices up to that bound. More precisely:

1. We prove that all hollow 4-simplices (in particular, empty ones) of width larger than two have determinant at most 7588. See details in Section 2.
2. We enumerate all empty 4-simplices of determinant ≤ 7600. See details en Section 3.

2 Bounding the volume of wide hollow 4-simplices

Throughout this section, P is a hollow 4-simplex of width three or more. In order to prove that $\det(P) \leq 7588$ (Theorem 7) we separate the cases when P projects to a hollow 3-polytope or not.

2.1 P admits a hollow projection

Suppose that there is an affine integer projection $\pi : P \to Q \subset \mathbb{R}^3$ with Q hollow. Then the width of Q is at least that of P (because any affine integer funcional on Q can be lifted to P, with the same width; still, P could have smaller width with respect to a functional that is not compatible with the projection).

Thus, Q is a hollow lattice 3-polytope of width at least three. Such polytopes have been classified and there are actually only five, all of width three [1, 3]. Thanks to the classification we can prove:

Theorem 5 *If an empty 4-simplex P of width at least three has a hollow lattice projection to dimension three then the determinant of P is at most 27.*

In this proof and in the rest of the paper, we call *normalized volume* of a full-dimensional lattice polytope $P \in \mathbb{R}^d$ its Euclidean volume multiplied by $d!$. For example, the normalized volume of a simplex equals (the absolute value of) its determinant. Since every lattice polytope can be triangulated and since the determinant of a lattice simplex is an integer, the normalized volume of every lattice polytope is an integer too. We also use the *lattice length* of a segment s with rational direction, defined as its length normalized to the distance between two consecutive lattice points in that direction.

Proof. (Sketch) We look one by one at the five possibilities for the hollow 3-polytope Q. They are:

- A triangular prism, which cannot be the projection of a 4-simplex since it has more than five vertices.
- Three tetrahedra, of normalized volumes 27, 25, and 27. If one of these equals Q, then a facet F of P already projects to Q and it projects bijectively. Let v be the vertex of P opposite to F and let s be the vertical line through v (we call vertical the projection direction). Then, the normalized volume of P equals the normalized volume of Q times the lattice length of the segment $P \cap s$. Since the line s is a lattice line (it contains v), the length of the lattice-free segment $P \cap s$ must be at most one. That is, the normalized volume of P is at most that of Q.
- One (combinatorially equivalent to a) square pyramid. Suppose Q is this one. Let o be the

intersection point of the two diagonals of the quadrilateral facet of Q (which happens to be a lattice point) and let s be the vertical line projecting to o. We skip details, but it is easy to show that the normalized volume of P again equals that of Q times the length of $P \cap s$. As before, since s is a lattice line this length is at most one, so the normalized volume of P is at most that of Q, which equals 27 again.

□

2.2 P does not admit a hollow projection

Here we need some tools from convex geometry. We start with Minkowski's Second Theorem about successive minima. The *i-th successive minimum* ($i = 1, \ldots, d$) of a centrally symmetric convex body $C \in \mathbb{R}^d$ is the smallest dilation factor λ_i such that $\lambda_i C$ contains i linearly independent lattice points. Minkowski's Second Theorem [4] says that for such a C one has:

$$\mathrm{vol}(C) \leq \frac{2^d}{\lambda_1 \lambda_2 \cdots \lambda_d}.$$

(Here the volume is meant *Euclidean*, not *normalized*. Remember that the latter equals $d!$ times the former).

An example of a centrally symmetric body is the difference body $K - K := \{a - b : a, b \in K\}$ of a convex body $K \subset \mathbb{R}^d$. It is easy to notice that the first successive minimum ${\lambda_1}^{-1}(K-K)$ equals the *lattice diameter* of K: the maximum lattice length of a rational segment contained in K. On the other hand we have that

$$\mathrm{vol}(K) \leq \frac{\mathrm{vol}(K-K)}{2^d} \leq \frac{1}{\lambda_1 \lambda_2 \cdots \lambda_d} \leq \frac{1}{{\lambda_1}^d}, \quad (1)$$

where the λ_i's are the successive minima of $K - K$. The first inequality is (a particular case of) the Brunn-Minkowski inequality [4], the second one is Minkowski's Second Theorem, and the third inequality comes from $\lambda_1 \leq \cdots \leq \lambda_d$ (which is obvious from the definition).

To proceed further, we need a bound on the maximum volume of hollow 3-polytopes of a certain width. The following generalizes a result from Averkov et al. [1, Proposition 11] (the original statement is only for $w = 3$).

Proposition 6 *Let $w > 1 + 2/\sqrt{3} = 2.155$ and let $\mu = w^{-1}$. Then, the following statements hold for any convex body $K \subset \mathbb{R}^3$ with no interior integer points and of lattice width at least w:*

(a) $\lambda_1(K - K) \geq 1 - (1 + 2/\sqrt{3})\mu$.

(b) $\mathrm{vol}(K)$ *is bounded above by*

$$\begin{cases} 3/(4\mu^2(1 - \mu(1 + 2/\sqrt{3}))) & \text{if} \quad w \leq 2.427, \\ 8/(1-\mu)^3 & \text{if} \quad w \geq 2.427. \end{cases}$$

Let now P be an empty 4-simplex of width at least three and assume it does not have a hollow lattice projection into dimension three. Let $Q = \pi(P)$ be the projection along the direction where λ_1 is achieved. By our hypotheses, Q is non-hollow and has width at least three.

Let $x \in Q$ be the projection point of the segment where λ_1 is achieved (so that if s_x denotes the vertical line through x, then $1/\lambda_1$ equals the length of the segment $P \cap s_x$). Let $y \in Q$ be an interior lattice point, which exists since Q is non-hollow, and consider the vertical line s_y through it. As in the proof of Theorem 5, we have that:

- The normalized volume of P equals that of Q times the length of $s_x \cap P$.
- The length of $s_y \cap P$ cannot be larger than one, since s_y is a lattice line.

The difficulty now is, of course, that x and y are (or may be) different points. But we can relate the lengths of $s_x \cap P$ and $s_y \cap P$ as follows: consider the ray from x through y and let z be the point where it hits the boundary of P. Then, we have:

$$\lambda_1(P - P)^{-1} = |s_x \cap P| \leq \frac{|s_x \cap P|}{|s_y \cap P|} \leq \frac{|xz|}{|yz|}.$$

So, any bound for $|xz|/|yz|$ is also a bound for ${\lambda_1}^{-1}$ and, in turn, a bound for $\mathrm{vol}(P)$ via Equation (1).

In order to find one such bound, assume that y is the interior lattice point where the quotient $|xz|/|yz|$ is as small as possible. Observe that $r := 1 - |xz|/|yz|$ equals the greatest dilation factor of an homothety of Q with center at x such that rQ is hollow. By what we said above,

$$\lambda_1 \geq \frac{|yx|}{|xz|} \geq 1 - r.$$

We skip details of the rest of the proof, but the general idea is that now:

- If ${\lambda_1}^{-1}$ is smaller than, say, 4, then Equation (1) already implies a bound of at most 256 for the Euclidean volume of P (which translates to $256 \times 24 \simeq 10,000$ for the normalized volume).
- If ${\lambda_1}^{-1}$ is greater than 4, then $1-r$ is smaller than $1/4$, so that r is greater than $3/4$. The width of rQ is r times the width of Q, so it is at least $9/4 = 2.25$. The hollow polytope rQ is then in the conditions of Proposition 6, which gives us an upper bound for the volume of it, hence for the volume of Q, and hence for the volume of P (for the latter we use arguments similar to those in the proof of Theorem 5).

By putting together these two facts (and optimizing the parameters a bit) we arrive to:

Theorem 7 *There is no hollow 4-simplex of width 3 with volume greater than 7588.*

3 Computations

For the computation of all empty 4-simplices of a given determinant D we have two different algorithms, depending on D. In both we obtain a list of perhaps-non-empty simplices that contains all the empty ones, and then prune via an emptyness test:

- (Algorithm 1) Every lattice 4-simplex that has a unimodular facet is equivalent to

  $$\Delta(v) := \operatorname{conv}\{e_1, e_2, e_3, e_4, v\}$$

  for some $v = (v_1, v_2, v_3, v_4) \in \mathbb{Z}^4$ with $\sum v_i = D + 1$. Moreover, v needs only to be considered modulo D, which gives a priori D^4 possibilities.

  This algorithm can be used for all values of D with less than five different prime factors, since the fact that empty 4-simplices are cyclic [2] easily implies that the determinants of different facets are relatively prime.

- (Algorithm 2) When $D = pq$ with p and q relatively prime, every 4-simplex Δ_D of determinant D can be obtained by "merging" simplices Δ_p and Δ_q of determinants p and q. Here, merging Δ_p and Δ_q means that Δ_p and Δ_q are realized as the same standard simplex, but considered with respect to superlattices Λ_p and Λ_q of $\mathbb{Z}^4$. Their merging is the same simplex, considered with respect to $\Lambda_p + \Lambda_q$. Since Λ_p and Λ_q refine $\mathbb{Z}^4$ with relatively prime indices p and q, $\Lambda_p + \Lambda_q$ refines $\mathbb{Z}^4$ with index pq. This procedure exhausts all empty 4-simplices of volume D because they are all cyclic (of order D) and $\mathbb{Z}_D \cong \mathbb{Z}_p \oplus \mathbb{Z}_q$.

For many values of D both algorithms apply. When this happens, experimentally we have found that Algorithm 2 is faster. Also, when there are choices on how to break D as a product pq with $\gcd(p, q) = 1$, the fastest results are obtained for p and q of about the same size. This is illustrated in Table 1, where the running time of the complete enumeration (and emptyness test) for some (approximate) values of D are shown. (The times shown are for a single D, and in algorithm 2 the lists of all empty 4-simplices of volumes p and q are considered precalculated).

The algortihms have been implemented in python and runned in the Altamira node of Spanish Supercomputing Network. The total CPU time needed to enumerate all empty 4-simplices up to determinant 7 600 was above 10 000 hours.

D	Algor. 1	Algor. 2 $p \simeq q$	Algor. 2 $p \ll q$
$\simeq 2000$	0.53	0.29	1.06
$\simeq 3000$	1.14	0.65	7.32
$\simeq 4000$	5.31	1.17	17.76
$\simeq 5000$	11.45	2.32	10.31
$\simeq 6000$	21.88	4.42	21.38
$\simeq 7000$	38.59	6.69	26.95

Table 1: Computation time (hrs.) for enumeration of all empty 4-simplices of a given determinant D

References

[1] G. Averkov, J. Krumplemann and S. Weltge. Notions of maximality for integral lattice-free polyhedra: the case of dimension three, arXiv:1509.05200

[2] M. Barile, D. Bernardi, A. Borisov and J.-M. Kantor. On empty lattice simplices in dimension 4, *Proc. Amer. Math. Soc.* 139(12):4247–4253, 2011.

[3] M. Blanco, C. Haase, J. Hoffman and F. Santos. The finiteness threshold width of lattice polytopes, arXiv:1607.00798v2

[4] P. M. Gruber. Geometry of numbers, Handbook of Convex Geometry, Vol. A, B, North-Holland, Amsterdam, 1993, pp. 739–763.

[5] C. Haase and G. M. Ziegler. On the maximal width of empty lattice simplices. *Europ. J. Combin.* 21 (2000), 111–119.

[6] J. M. Kantor and K. S. Sarkaria. On primitive subdivisions of an elementary tetrahedron. *Pacific J. Math.*, 211(1):123–155, 2003.

[7] S. Mori, D. R. Morrison and I. Morrison. On four-dimensional terminal quotient singularities. *Math. Comput.* 51 (1988), no. 184, 769-786.

[8] G. Pick. Geometrisches zur Zahlenlehre, *Sonderabdr. Naturw.-medizin. Verein f. Böhmen "Lotos"*, 19:311–319, 1899.

[9] F. Santos and G. M. Ziegler. Unimodular triangulations of dilated 3-polytopes. *Trans. Moscow Math. Soc.*, 74 (2013), 293–311.

[10] A. Sebő. An introduction to empty lattice simplices, in *Integer Programming and Combinatorial Optimization (Graz 1999)*, Lecture Notes in Comput. Sci. 1610, Springer-Verlag, Berlin, 1999, pp. 400–414.

[11] G. K. White. Lattice tetrahedra. *Canadian J. Math.* 16 (1964), 389–396.

Convex Quadrangulations of Bichromatic Point Sets

Alexander Pilz* Carlos Seara†

Abstract

We consider quadrangulations of red and blue points in the plane where each face is convex and no edge connects two points of the same color. In particular, we show that the following problem is NP-hard: *Given a finite set S of points with each point either red or blue, does there exist a convex quadrangulation of S in such a way that the predefined colors give a valid vertex 2-coloring of the quadrangulation?* We consider this as a step towards solving the corresponding long-standing open problem on monochromatic point sets.

1 Introduction

A *quadrangulation* of a set S of n points in the Euclidean plane is a partition of the convex hull of S (denoted by $\mathsf{CH}(S)$) into quadrangles (i.e., 4-gons) such that the union of the vertices of the quadrangles is exactly the point set S and two quadrangles are disjoint or intersect either in a common vertex or a common edge. Hence, the quadrangulation is also a geometric (straight-line) planar graph with vertex set S. A quadrangulation is a *convex quadrangulation* if every quadrangle is convex. A point set admits a quadrangulation if and only if the number of points on the convex hull is even [3], but not every such set admits a convex quadrangulation, and deciding this in polynomial time is an open problem (posed by Joe Mitchell already in 1993 [15]).

A graph is *vertex k-colorable* (in brief *k-colorable*) if there exists a mapping of each vertex of the graph to exactly one of k colors such that no two vertices of the same color share an edge. A 2-colorable graph is a *bipartite graph*. It is known that every quadrangulation is bipartite. A *bichromatic point set* is a finite set S of points together with a mapping of each point to one of two colors. Throughout this paper, these colors will be *red* and *blue*.

Our main question is whether for a given bichromatic point set there is a convex quadrangulation s.t. the colors of the points define a valid 2-coloring of the quadrangulation. We call such a quadrangulation *valid*. Consider a 2-coloring of any quadrangulation. There are at least two vertices of each color, and it is easy to construct examples of quadrangulations with any valid number of vertices that have only two vertices of one color.

In Section 2, we prove that this bound differs for convex quadrangulations. Using observations of this section, we show that deciding whether a bichromatic point set has a valid convex quadrangulation is NP-complete. Mitchell's motivating question is left open.

Quadrangulations. Quadrangulations of point sets or polygons were discussed by many authors; see the survey by Toussaint [15]. Since not all polygons or point sets admit quadrangulations, even when the quadrangles are not required to be convex, the author surveys results on the characterization of those planar sets that always admit quadrangulations (convex and non-convex): quadrangulations of orthogonal polygons, simple polygons and point sets.

Lubiw [12] shows that determining whether a simple polygon with holes has a convex quadrangulation is NP-complete, even when quadrangles of any form are allowed; in contrast to that, there is a polynomial-time algorithm for a generalized variant of rectilinear polygons. Bose and Toussaint [3] show that a set S of n points admits a quadrangulation if and only if S has an even number of extreme points. They present an algorithm that computes a quadrangulation of S in $O(n \log n)$ time even in the presence of collinear points, adding an extra extreme point if necessary. Ramaswami, Ramos, and Toussaint [13] present efficient algorithms for converting triangulated domains to quadrangulations, while giving bounds on the number of Steiner points that might be required to obtain the quadrangulations. They show that a triangulated simple n-gon can be quadrangulated in linear time with the least number of outer Steiner points required for that triangulation, and that $\lfloor \frac{n}{3} \rfloor$ outer Steiner points are sufficient, and sometimes necessary, to quadrangulate a triangulated simple n-gon. They further show that $\lfloor \frac{n}{4} \rfloor$ inner Steiner points (and at most one outer Steiner point) are sufficient to quadrangulate a triangulated simple n-gon, and this can be done in linear time. The method can be used to quadrangulate arbitrary triangulated domains.

*Department of Computer Science, ETH Zürich, alexander.pilz@inf.ethz.ch. Supported by a Schrödinger fellowship, Austrian Science Fund (FWF): J-3847-N35.

†Departament de Matemàtiques, Universitat Politècnica de Catalunya, carlos.seara@upc.edu. Supported by the projects Gen. Cat. DGR 2014SGR46, MINECO MTM2015-63791-R, and the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 734922.

Convex quadrangulations. Most of the work on convex quadrangulations is concerned with Steiner points. For example, Bremner et al. [4] prove that if the convex hull of S has an even number of points, then by adding at most $\frac{3n}{2}$ Steiner points in the interior of its convex hull, we can always obtain a point set that admits a convex quadrangulation. The authors also show that $\frac{n}{4}$ Steiner points are sometimes necessary. Heredia and Urrutia [8] improve these upper and lower bounds to $\frac{4n}{5}+2$ and $\frac{n}{3}$, respectively. Deciding in polynomial time whether a given (monochromatic) point set admits a convex quadrangulation without adding Steiner points seems to be a long-standing open problem. Only fixed-parameter-tractable algorithms and heuristics are known. Fevens, Meijer, and Rappaport [7] present a polynomial-time algorithm to determine whether a point set S admits a convex quadrangulation if S is constrained to lie on a constant number of nested convex polygons. Schiffer, Aurenhammer, and Demuth [14] propose a simple heuristic for computing large subsets of convex quadrangulations on a given set of points in the plane.

Quadrangulations of colored point sets. Cortés et al. [6] discuss aspects of quadrangulations of bichromatic point sets. They study bichromatic point sets that admit a quadrangulation, and whether, given two quadrangulations of the same bichromatic point set, it is possible to transform one into the other using certain local operations. They show that any bichromatic point set with convex layers having an even number of points with alternate colors has a valid quadrangulation, and any two such quadrangulations can be transformed into each other.

Alvarez, Sakai, and Urrutia [2] prove that a bichromatic set of n points can be quadrangulated by adding at most $\lfloor\frac{n-1}{3}\rfloor+\lfloor\frac{n}{2}\rfloor+1$ Steiner points and that $\frac{m}{3}$ Steiner points are occasionally necessary, where m is the number of quadrilaterals of the quadrangulation. They also show that there are 3-colored point sets with an even number of extreme points that do not admit a quadrangulation, even after adding Steiner points inside the set's convex hull.

Kato, Mori, and Nakamoto [9] define the *winding number* $\omega(S)$ for a 3-colored point set S, and prove that a 3-colored set S of n points in general position with a finite set P of Steiner points added is quadrangulatable if and only if $\omega(S)=0$. When $S\cup P$ is quadrangulatable, then $|P|\leq\frac{7n+34m-48}{18}$, where the number of extreme points is $2m$. This line of research is continued by Alvarez and Nakamoto [1], who study k-colored quadrangulation of k-colored sets of points, where $k\geq 2$. They show that if $\omega(S)=0$ or $k\geq 4$, then a k-colored quadrangulation of S can always be constructed using less than $\frac{(16k-2)n+7k-2}{39k-6}$ Steiner points. (We note that $\omega(S)=0$ for any bichromatic S where red and blue points on $\mathsf{CH}(S)$ alternate.)

2 The red and the blue graph of a convex quadrangulation

Let Q be a convex quadrangulation with a valid red-blue coloring of its n vertices. For every quadrangle, one diagonal connects the two red vertices of the quadrangle, and the other connects the two blue ones. We call them the *red diagonal* and the *blue diagonal*, respectively. Let G_{R} be the graph whose vertices are the red vertices of Q and whose edges are the red diagonals of all quadrangles of Q. Let G_{B} be defined analogously; since the colors are interchangeable, all the following statements hold equally for both graphs. Since every red edge has its own quadrangle and the faces (quadrangles) are convex, we obtain the following results.

Observation 1 *G_{R} is a plane simple graph.*

The following lemma is proven in the full version.

Lemma 2 *G_{R} is connected.*

Lemma 3 *Every minimal cycle of G_{R} contains exactly one blue point in its interior, and every inner blue point is contained in a minimal cycle of G_{R}. Blue points on the convex hull boundary are separated from the remaining set by a path in G_{R}.*

Proof. Consider the quadrangles that are adjacent to an inner blue point. The red diagonals of the quadrangles form a cycle that contain the blue point. Further, consider any minimal cycle of G_{R} and any edge therein. This edge corresponds to a quadrangle and there is one blue point of the quadrangle on each side of the edge. Observe that, in the same way, every blue point on the convex hull boundary is separated by a red path from the other blue vertices. □

Theorem 4 *Let n_{R} and n_{B} be the number of red and blue vertices, respectively, of a 2-colored convex quadrangulation. Then $n_{\mathrm{B}}\leq 2n_{\mathrm{R}}-2$.*

Proof. Observe that G_{R} and G_{B} have the same number e of edges. By Euler's Polyhedral Formula we have $n_{\mathrm{B}}-e+f_{\mathrm{B}}=2$, where f_{B} is the number of faces in the blue graph (including the outer face). Lemma 3 implies $n_{\mathrm{R}}=f_{\mathrm{B}}-1+\frac{h}{2}$, where h is the number of points in the convex hull. Hence, we get $n_{\mathrm{R}}+n_{\mathrm{R}}-\frac{h}{2}-1=e$. Since G_{R} is a plane geometric graph, we have $e\leq 3n_{\mathrm{R}}-3-\frac{h}{2}$. By plugging this into the previous equation we get the claimed inequality. □

Note that the inequality $e\leq n_{\mathrm{R}}-3-\frac{h}{2}$ is tight if and only if G_{R} is a triangulation. Figure 1 shows an example where the bound is tight.

The structure of the red and the blue graph reveals a necessary condition of a bichromatic point set

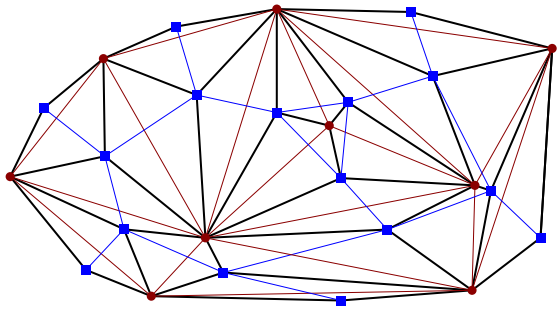


Figure 1: An example showing that the bound on the relation between the red (round) and blue (squared) points in a convex quadrangulation (indicated by thick black segments) is tight.

that allows a convex quadrangulation: Every segment between two red points must be intersected by a segment between two blue points. (Cortés et al. [5] give a quadratic-time algorithm to check for this property.)

3 NP-completeness

In this section we prove that the problem of deciding whether there exists a valid convex quadrangulation of a given bichromatic set of points is NP-hard.

Our reduction is from planar 3-SAT (cf. [11]). The construction is based in large parts on placing two red points sufficiently close to a crossing between two segments between blue points, s.t. exactly one of these blue segments is a diagonal of a quadrilateral in any convex quadrangulation (recall Lemma 3), and that the state of variables is propagated between the gadgets. Once there is a valid choice of these blue diagonals (corresponding to a satisfying variable assignment), we need to show that they are part of a valid convex quadrangulation. We later argue that the construction is possible with coordinates of polynomial size.

As common in this type of reductions, we transform an embedding of a planar 3-SAT instance to a bichromatic point set by replacing elements of the graph drawing by gadgets. For simplicity, we may consider the drawing to consist of edges that are represented by a sequence of orthogonal line segments (actually, one bend suffices, see [10]). An edge in this graph carries the truth value of a variable to the clause gadgets (possibly via a negation).

The main part of an edge gadget consists of a chain of *link gadgets*, each containing four blue points in convex position and two red points close to the crossing they define. Hence, one of the two blue edges must be a diagonal in any valid convex quadrangulation Q (if it exists). See Figure 2. If one of the segments is a diagonal of Q (say, the one from bottom-left to top-right), the edge gadget carries *true* (and the line segment is called the *T-diagonal* of the link gadget);

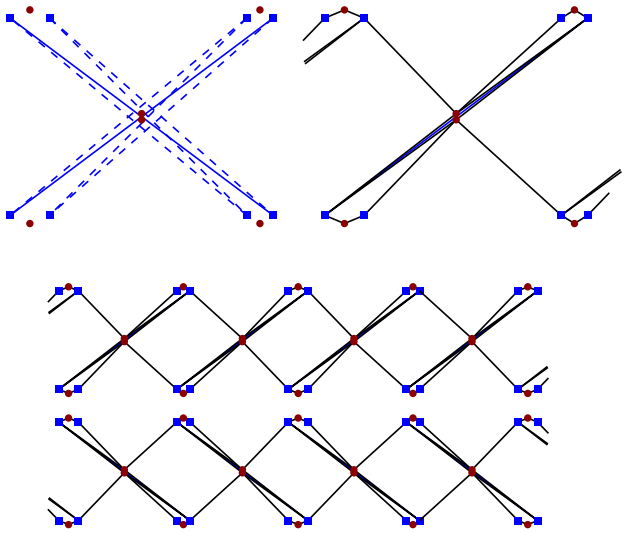


Figure 2: A link gadget to model edges in the graph. The middle red points are that close to the crossing of the solid blue segments s.t. there is no other segment passing between them (as indicated by the dashed lines). (Note that to this end, the three-point "caps" at the ends of the segments have to have slightly different width.) Exactly one of the blue segments has to be a diagonal of the quadrangulation, and combining these links propagates that decision. A possible quadrangulation is shown to the right. The link gadgets can be concatenated to form edges, as shown below.

if the other segment (being called the *F-diagonal*) is a diagonal of Q, the edge gadget carries *false*. Two of these links are joined such that the T-diagonal of the previous link crosses the F-diagonal of the next link, and vice versa, and thus Q cannot have a T-diagonal and an F-diagonal in the same edge gadget.

A variable gadget works by connecting three edge gadgets in a way that they all have either the T-edge or the F-edge as a diagonal; an arbitrary number of edges from the same variable vertex can be connected in that way. The variable gadget is shown in Figure 3. Further, we need bends in the edge gadgets to connect horizontal and vertical parts, as well as negation gadgets. All of these are mere appropriate combinations of link gadgets, figures and exact descriptions of these gadgets are provided in the full version.

For the clause gadgets, we have a pair of red points that span a segment intersected by exactly three potential blue diagonals. There is only one state of the three incident edge gadgets that prevents all three blue diagonals. By appropriately adding negation gadgets, we make this configuration appear exactly when all three edge gadgets carry *false*. See Figure 4.

It remains to argue that the parts of the convex hull not covered by gadgets can be quadrangulated. As these parts are enclosed by a polygonal chain in which red and blue vertices alternate, we can add

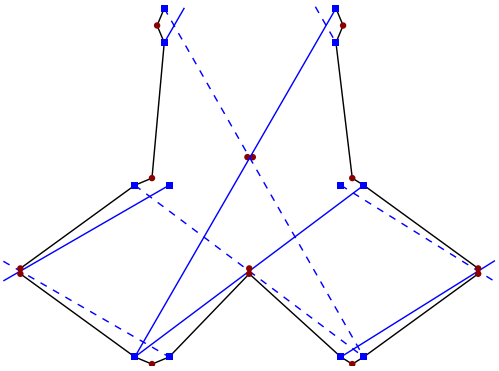


Figure 3: A variable gadget, showing the possible set of blue diagonals. It "splits" an edge, propagating the (negated) truth value it carries.

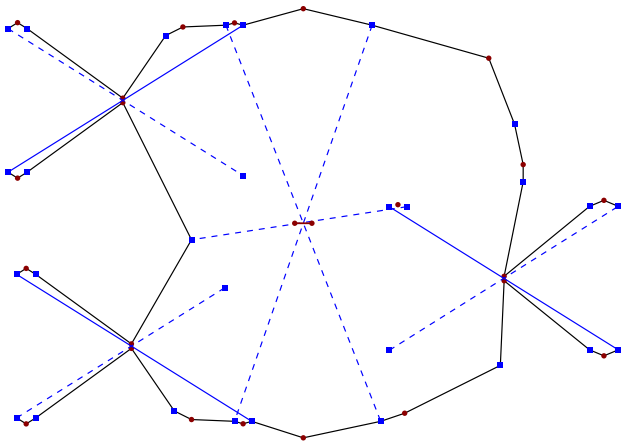


Figure 4: The clause gadget. The two red points in the middle connected by a red segment are closer than drawn. The only impossible combination of blue diagonals for the link gadgets is the one including all three solid segments. Negating the top-left edge makes this the configuration with all literals set to *false*.

Steiner points to find a quadrangulation (see the full version for details). Thus, after adding these Steiner points to our construction, there is a bichromatic convex quadrangulation of our point set if and only if the corresponding planar 3-SAT instance is satisfiable.

Finally, let us remark that the points can be placed in general position using coordinates of polynomial size. Before placing two close red points, we find out which distance allows placing the points sufficiently close to each other (possibly after a perturbation).

Theorem 5 *Given a set of red and blue points in the plane, it is NP-complete to decide whether there is a valid convex quadrangulation of that point set.*

Note that actually, the bichromatic setting is a way to forbid certain edges in the quadrangulation. For our reduction, it is sufficient to forbid those between the close red points in the gadgets. However, we do not know how to achieve this in an unconstrained setting.

References

[1] V. Alvarez and A. Nakamoto. Colored quadrangulations with Steiner points. In J. Akiyama, M. Kano, and T. Sakai, editors, *TJJCCGG 2012*, volume 8296 of *LNCS*, pages 20–29. Springer, 2012.

[2] V. Alvarez, T. Sakai, and J. Urrutia. Bichromatic quadrangulations with Steiner points. *Graphs Combin.*, 23:85–98, 2007.

[3] P. Bose and G. Toussaint. Characterizing and efficiently computing quadrangulations of planar point sets. *Comput. Aided Geom. Des.*, 14(8):763 – 785, 1997.

[4] D. Bremner, F. Hurtado, S. Ramaswami, and V. Sacristán. Small strictly convex quadrilateral meshes of point sets. *Algorithmica*, 38(2):317–339, 2004.

[5] C. Cortés, D. Garijo, M. A. Garrido, C. I. Grima, A. Márquez, A. Moreno-González, J. Valenzuela, and M. T. Villar. Reporting bichromatic segment intersections from point sets. *Int. J. Comput. Geometry Appl.*, 22(5):421–438, 2012.

[6] C. Cortés, A. Márquez, A. Nakamoto, and J. Valenzuela. Quadrangulations and 2-colorations. In *Proc. 21st EuroCG*, pages 65–68, 2005.

[7] T. Fevens, H. Meijer, and D. Rappaport. Minimum convex partition of a constrained point set. *Discrete Appl. Math.*, 109(1–2):95 – 107, 2001.

[8] V. M. Heredia and J. Urrutia. On convex quadrangulations of point sets on the plane. In *CJCDGCGT*, volume 4381 of *LNCS*, pages 38–46, 2005.

[9] S. Kato, R. Mori, and A. Nakamoto. Quadrangulations on 3-colored point sets with Steiner points and their winding numbers. *Graphs Combin.*, 30(5):1193–1205, 2014.

[10] D. E. Knuth and A. Raghunathan. The problem of compatible representatives. *SIAM J. Discret. Math.*, 5(3):422–427, 1992.

[11] D. Lichtenstein. Planar formulae and their uses. *SIAM J. Comput.*, 11(2):329–343, 1982.

[12] A. Lubiw. Decomposing polygonal regions into convex quadrilaterals. In *Proc. 1st SoCG*, pages 97–106, 1985.

[13] S. Ramaswami, P. Ramos, and G. Toussaint. Converting triangulations to quadrangulations. *Comput. Geom.*, 9(4):257 – 276, 1998.

[14] T. Schiffer, F. Aurenhammer, and M. Demuth. Computing convex quadrangulations. *Discrete Appl. Math.*, 160(4–5):648 – 656, 2012.

[15] G. T. Toussaint. Quadrangulations of planar sets. In *WADS*, volume 955 of *LNCS*, pages 218–227, 1995.

Perfect k-colored matchings and $k+2$-gonal tilings

Oswin Aichholzer* Lukas Andritsch† Karin Baur† Birgit Vogtenhuber*

Abstract

We derive a simple bijection between geometric plane perfect matchings on $2n$ points in convex position and triangulations on $n+2$ points in convex position. We then extend this bijection to monochromatic plane perfect matchings on periodically k-colored vertices and $(k+2)$-gonal tilings of convex point sets. These structures are related to Temperley-Lieb algebras and our bijections provide explicit one-to-one relations between matchings and tilings. Moreover, for a given element of one class, the corresponding element of the other class can be computed in linear time.

1 Introduction

The Fuss-Catalan numbers $f(k,m) = \frac{1}{m}\binom{km+m}{m-1}$ are known to count the number of $k+2$-gonal tilings of a convex polygon of size $km+2$, they go back to Fuss-Euler (cf. [4]). Bisch and Jones introduced k-colored Temperley-Lieb algebras in [1] as a natural generalisation of Temperley-Lieb algebras. These algebras have representations by certain planar k-colored diagrams with $m(k+1)$ vertices on top and bottom. The dimension of such an algebra is $f(k,m)$, with a basis indexed by these diagrams. We call these diagrams plane perfect k-colored matchings or just k-colored matchings, assuming from now on that they are plane and perfect. Since the number of $k+2$-gonal tilings coincides with the number of k-colored matchings, these sets are in bijection. Przytycki and Sikora [4] prove this through an inductive implicit construction but do not give an explicit bijection of the structures.

Furthermore, from work of Marsh and Martin [3], one can derive an implicit correspondence between triangulations and diagrams for $k=1$. However, to our knowledge, no explicit bijection is known.

In this paper, we will give bijections between these two sets of plane graphs on sets of points in convex position. We will first address the case $k=1$ (Section 2) and then treat the general case. Our main theorems are the explicit bijections between the set of k-colored matchings and the $(k+2)$-gonal tilings (Theorems 1 and 8). A key ingredient is the characterization of valid k-colored matchings in Theorem 3. Due to lack of space, most proofs are deferred to the full version of this paper.

2 Matchings and triangulations

We will draw the matchings with two parallel rows of n vertices each, labeled v_1 to v_n and v_{n+1} to v_{2n} in clockwise order, and with non-straight edges; see Figure 1(left). We will draw the triangulations (and tilings) on $n+2$ points in convex position, labeled p_1 to p_{n+2} in clockwise order; see Figure 1(right). For the sake of distinguishability, throughout this paper we will refer to $p_1,\dots,p_{n+2}$ as *points* and to $v_1,\dots,v_{2n}$ as *vertices*.

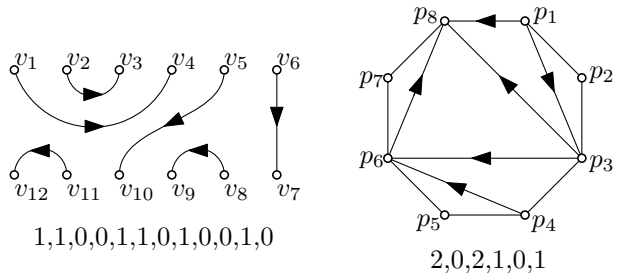


Figure 1: A perfect matching (left) and the corresponding triangulation for $n=6$ (right).

The above defined structures are undirected graphs. We next give an implicit direction to the edges of these graphs: an edge v_iv_j (p_ip_j) is directed from v_i to v_j (p_i to p_j) for $i<j$, that is, each edge is directed from the vertex / point with lower index to the vertex / point with higher index. This also defines the outdegree of every vertex / point, which we denote as b_i for each vertex v_i and as d_i for each point p_i. For technical reasons, we do not count the edges of the convex hull of a triangulation when computing the outdegree of a point p_i, with the exception of the edge p_1p_{n+2}. We call the sequence $(b_1,\dots,b_{2n})$ of the outdegrees of a matching (or the sequence $(d_1,\dots,d_n)$ of the first n outdegrees of a triangulation) its *outdegree sequence*; see again Figure 1. We first show that for both structures, this sequence is sufficient to encode the graph.

For matchings, the outdegree sequence is a 0/1-sequence with $2n$ digits, where n digits are 1 and n digits are 0. Moreover, the directions of the edges imply that an incoming edge at a vertex v_j must be outgoing for a vertex v_i with $i<j$. Thus, we have the condition $\sum_{i=1}^{k} b_i \geq k/2$ for any $1 \leq k \leq 2n$, that is, in any subsequence starting at v_1, we have

*Institute for Software Technology, Graz University of Technology, Graz, Austria, [oaich|bvogt]@ist.tugraz.at

†Mathematics and Scientific Computing, University of Graz, Graz, Austria, [baurk|lukas.andritsch]@uni-graz.at

at least as many 1s as 0s. Such sequences are called ballot sequences, see [2, p.69]. Obviously, the outdegree sequence of a matching can be computed from a given matching in $O(n)$ time. But also the reverse is true: We consider the outdegrees from b_1 to b_{2n}. We use a stack (with the usual push and pop operations) to store the indices of considered vertices that still need to be processed. Initially, the stack is empty. If $b_i = 1$, we push the index i on the stack. If $b_i = 0$, we pop the topmost index k from the stack and output the edge $v_k v_i$. In this way, always the last vertex with 'open' outgoing edge is connected to the next vertex with incoming edge, implying that the subgraph with vertices v_k to v_i is a valid plane perfect matching. A simple induction argument shows that the whole resulting graph is plane and can be reconstructed from the outdegree sequence in $O(n)$ time.

For triangulations, first note that the outdegrees of p_{n+1} and p_{n+2} are 0. Thus we do not lose information when restricting the outdegree sequence of a triangulation to $(d_1, \ldots, d_n)$. Similar as before, the directions of edges imply that for any valid outdegree sequence, it holds that $\sum_{i=1}^{k} d_{n+1-i} \leq \sum_{i=1}^{k} 1 = k$ for any $1 \leq k \leq n$. This sum is precisely the maximum number of edges which can be outgoing from the 'last' k points p_{n+1-k} to p_n. Recall that we do not consider the edges of the convex hull, except for $p_1 p_{n+2}$, and thus the number of edges which contribute to the outdegree sequence is exactly $n-2$. As before, it is straightforward to compute the outdegree sequence from a given triangulation in $O(n)$ time. For the reverse process, we again use a stack to store the indices of considered points that still need to be processed. We initialize the stack with push$(n+2)$ and push$(n+1)$ and output all the (non-counted) edges $p_i p_{i+1}$ for $1 \leq i \leq n+1$. Then we consider the outdegrees in reversed order, that is, from d_n to d_1. For each degree d_i we perform two steps. (1) d_i times, we pop the topmost index from the stack and after each pop we output the edge $p_i p_k$, where k is the (new) topmost index on the stack. (2) We push i on the stack. This process constructs the triangulation from back to front. When processing p_i, all points from p_{i+1} to p_{n+2} that are still 'visible' from p_i are in this order on the stack. Thus, drawing the edges in the described way generates a planar triangulation. At the end of the process, the stack contains exactly the two indices $n+2$ and 1, which can be ignored.

So far we have shown that there exist one-to-one relations between outdegree sequences on the one side and matchings respectively triangulations on the other side. We now present a bijective transform between outdegree sequences of matchings and those of triangulations.

For a given outdegree sequence $B = (b_1, \ldots, b_{2n})$ of a perfect matching, we compute the outdegree d_i for the corresponding point of the triangulation as the number of 1s between the $(i-1)$-st 0 and the i-th 0 in B for $i > 1$, and set d_1 to the number of 1s before the first 0 in B.

For the reverse transformation, we process the outdegree sequence $(d_1, \ldots, d_n)$ of a triangulation from d_1 to d_n and set the entries of B in order from b_1 to b_n in the following way: For each entry d_i we first set the next d_i consecutive elements (possibly none) of B to 1; then we set the next element of B to 0.

It is an easy excercise to see that the two transformations are inverse to each other, and that they form a bijection between valid outdegree sequences of triangulations and outdegree sequences of matchings. Moreover, each transformation can be performed in $O(n)$ time. Figure 2 shows all corresponding perfect matchings, triangulations, and outdegree sequences for $n = 3$.

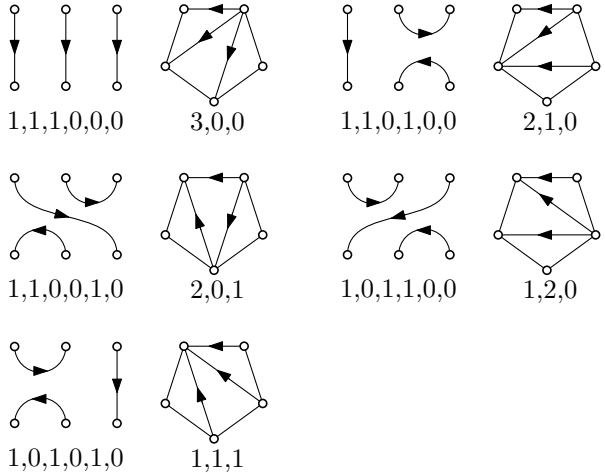


Figure 2: All perfect matchings, triangulations, and outdegree sequences for $n = 3$.

Theorem 1 *There exists a bijection between geometric plane perfect matchings on $2n$ points in convex position and geometric triangulations on $n+2$ points in convex position. Further, for an element of one structure, the corresponding element of the other structure can be computed in linear time.*

3 k-colored matchings

In this section we add colors to the vertices of the perfect matchings and require the matching edges to be monochromatic. For $k \geq 2$, let $c_1, \ldots, c_k$ be the k colors. We color the vertices in a bitonic way, that is, in the order $c_1, c_2, \ldots, c_{k-1}, c_k, c_k, c_{k-1}, \ldots, c_2, c_1, c_1, c_2, \ldots$ and so on. In a *perfect k-colored matching*, all matching edges connect vertices of the same color, and hence n is a multiple of k; see Figure 3 for an example of a k-colored matching with $k = 3$ colors and $n = 9$.

Clearly, any k-colored matching fulfills all conditions of the previous section. But not every match-

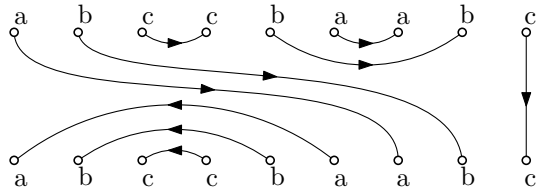


Figure 3: Perfect k-colored matching for $k = 3$ colors and $n = 9$ and its outdegree sequence.

ing obtained in the previous section is a k-colored matching and hence not every outdegree sequence of a matching is an outdegree sequence of a valid k-colored matching. Thus we now derive additional properties to determine which outdegree sequences of matchings correspond to k-colored matchings.

We denote k consecutive vertices $v_i, \ldots, v_{i+k-1}$ that are colored with either $c_1, \ldots, c_k$ or $c_k, \ldots, c_1$ as a *block*. In total we have $2n/k$ such blocks and they form a partition of $2n$ vertices. Observe that within a block, there cannot be a vertex with an incoming edge after a vertex with an outgoing edge, as this would cause a bichromatic edge. Hence, in a k-colored matching, the outdegree sequence of any block has to be of the form $|0, \ldots, 0, 1, \ldots, 1|$ (where it can consist entirely of 0 or 1 entries). For better readability, we sometimes mark block boundaries in an outdegree sequence with vertical lines. We say that an outdegree sequence (and the matching) fulfilling this property has a *valid block structure*.

Lemma 2 *Let M be a perfect matching with valid block structure that is not a k-colored matching. Then there exists an edge $v_s v_e$ in M with the following properties:*

(i) The vertices v_s and v_e lie in different blocks, say $v_s \in S$ and $v_e \in E$.

(ii) The subsequence from v_{s+1} to v_{e-1} contains no bichromatic matching edge.

(iii) The number of blocks between S and E is odd.

(iv) Let v_s be the i-th vertex in S. Then v_e is the $(i+1)$-st vertex in E.

Together with the previous observations, Lemma 2 implies the following theorem.

Theorem 3 *A matching is a k-colored matching if and only if it has a valid block structure and does not contain an edge as described in Lemma 2.*

Remark: For a given outdegree sequence we can check in linear time if it is an outdegree sequence of a k-colored matching by using the reconstruction algorithm described in Section 2.

4 t-gonal tilings

For any $t \geq 3$, a *t-gonal tiling* T on $n+2$ points in convex position, labeled p_1 to p_{n+2} in clockwise order, is a plane graph where every bounded face is a t-gon and the vertices along the unbounded face are $p_1, p_2, \ldots, p_{n+2}$ in this order; see Figure 4 for an example. For the special case of $t = 3$, T is a triangulation. In the next section, we will show that the k-colored matchings on $2n$ vertices of the previous section correspond to $k+2$-gonal tilings of $n+2$ points in convex position, where $n = km$ for some integer $m > 0$. This is a generalization of the fact that matchings (i.e., $k = 1$) correspond to triangulations. To this end we first derive several properties of t-gonal tilings of convex sets.

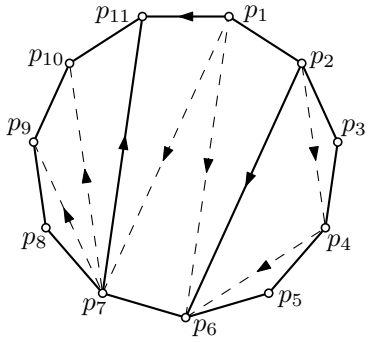


Figure 4: 5-gonal tiling corresponding to the 3-colored matching of Figure 3 and the outdegree sequence of its k-color valid triangulation.

The *dual graph* of a t-gonal tiling T has a vertex for each bounded face T and two vertices are connected by an edge if the corresponding faces share a common edge in T (every pair of bounded faces shares at most one edge). An *ear* of T is a t-gon which shares all but one edge with the unbounded face and can thus be cut off of T (along this edge) so that the remaining part is a valid t-gonal tiling of $n+2-(t-2) = n+4-t$ points.

As the dual graph of any t-gonal tiling T is a tree, as every tree has at least two leaves (where the minimal case is obtained by a path), and as a leaf in the dual graph of T corresponds to an ear in T, we have the following observation.

Observation 1 *Every t-gonal tiling with at least $2t-2$ points has at least two ears. At least one of these ears is not incident to the edge $p_1 p_{n+2}$.*

Lemma 4 *Any triangulation $\mathcal{T}$ on $n+2$ points in convex position contains at most one t-gonal tiling as a subgraph.*

A proof by induction, using Observation 1 can be found in the full version of this paper.

Obviously, if a triangulation $\mathcal{T}$ on $n+2$ points contains a t-gonal tiling T as a subgraph, then n is divisible by $t-2$. Further, as T has at least two ears, $\mathcal{T}$ contains at least two edges that cut off a triangulated t-gon from $\mathcal{T}$. We call such a t-gon that can be split off from a triangulation $\mathcal{T}$ a *t-ear* of $\mathcal{T}$, and the edge along which the t-ear can be split off the *ear-edge* (of the t-ear). Note that for $t > 3$, not every triangulation contains t-ears.

Let $\mathcal{T}$ be a triangulation that contains a t-ear with ear-edge p_rp_s for some $r \geq 1$ and $s = r+t-1 \leq n+2$. Let B be the outdegree sequence of the corresponding matching. If $s < n+2$, then in B, the t-ear corresponds to a subsequence W of B of length $2t-3$ that starts with a 1 (for p_rp_s), ends with two 0s (as the last point p_{s-1} of the ear cannot have outgoing edges), and has $t-1$ 0s and $t-2$ 1s in total. If $s = n+2$, then in B, the last 0 (the one 'after' p_{n+1}) is not existing. Then the according sequence is $W = (b_{2n-2t+5}, \ldots, b_{2n})$, which must be a ballot sequence.

5 k-colored matchings and $k+2$-gonal tilings

We say that a triangulation on $n+2$ points in convex position is *k-color valid* if it corresponds to a k-colored matching as defined in Section 3. The outdegree sequence of such a triangulation is then also called k-color valid. A $k+2$-gonal tiling of $n+2$ points is called k-color valid if it can be completed to (i.e., is a subgraph of) a k-color valid triangulation. In the following, let $t = k+2$.

Observation 2 *Let $\mathcal{T}$ be a k-color valid triangulation that contains a t-ear with ear-edge p_rp_s for some $r \geq 1$ and $s = r+t-1 \leq n+2$. Let the first entry of the subsequence W of B that corresponds to this t-ear be the i-th entry within its block, for $1 \leq i \leq k$. If $s = n+2$ then $i = 1$ and $W = (|1, \ldots, 1|0, \ldots, 0|) = (|1^k|0^k|)$. Otherwise, $W = (1, \ldots, 1|0, \ldots, 0, 1, \ldots, 1|0, \ldots, 0) = (1^{k-i+1}|0^{k-i+1}, 1^{i-1}|0^i)$.*

The following three lemmas can be derived using Observation 2. The proof of Lemma 5 also shows that the extension is uniquely determined.

Lemma 5 *Any k-color valid t-gonal tiling T on $n+2$ points can be extended by an ear at any edge $e = p_rp_{r+1}$, $1 \leq r \leq n+1$, so that the resulting t-gonal tiling on $n+k$ points is k-color valid.*

Lemma 6 *Let $\mathcal{T}$ be a k-color valid triangulation that contains a t-ear with ear-edge p_rp_s for some $r \geq 1$ and $s = r+t-1 \leq n+2$. Then the triangulation $\mathcal{T}'$ that results from removing the t-ear from $\mathcal{T}$ is again k-color valid.*

Lemma 7 *Let $\mathcal{T}$ be a k-color valid triangulation. Then $\mathcal{T}$ contains a t-ear with ear-edge p_rp_s for some $r \geq 1$ and $s = r+t-1 \leq n+2$.*

Combining Lemmas 4 – 7 and Observations 1 – 2, we obtain our main result.

Theorem 8 *There exists a bijection between geometric plane perfect k-colored matchings on $2n$ points in convex position and t-gonal tilings on $n+2$ points in convex position. Further, for an element of one structure, the corresponding element of the other structure can be computed in linear time.*

6 Future Work

The Temperley-Lieb algebras arising from matchings on $2n$ vertices can be generated by n distinguished elements: An element I (consisting of n propagating lines v_jv_{2n-j+1}, $1 \leq j \leq n$, from top to bottom) and $n-1$ elements U_i, $1 \leq i < n$, consisting of a pair of lines v_iv_{i+1} and $v_{2n-i}v_{2n-i+1}$ plus the remaining $n-2$ propagating lines.

It is natural to search for a characterization of these generators in terms of triangulations (and for the generators for the k-colored Temperley-Lieb algebras in terms of k+2-gonal tilings). We plan to use our explicit bijections to study the effect of edge flips in triangulations respectively in tilings on the corresponding matchings and to find out how the actions of generators of the (k-colored) Temperley-Lieb algebra can be interpreted in terms of flips in triangulations respectively in tilings. Preliminary results have already been obtained.

Acknowledgements. Research for this work is supported by the Austrian Science Fund (FWF) grant W1230. We thank Paul Martin for bringing this problem to our attention.

References

[1] D. Bisch and V. Jones. Algebras associated to intermediate subfactors. *Invent. Math.*, 128(1):89–157, 1997.

[2] W. Feller. *An Introduction to Probability Theory and its Applications, Volume I (3rd ed.)*. Wiley, 1968.

[3] R. J. Marsh and P. Martin. Pascal arrays: counting Catalan sets. *ArXiv Mathematics e-prints*, Dec. 2006.

[4] J. H. Przytycki and A. S. Sikora. Polygon dissections and Euler, Fuss, Kirkman, and Cayley numbers. *J. Combin. Theory Ser. A*, 92(1):68–76, 2000.

A Proof of the Orbit Conjecture for Flipping Edge-Labelled Triangulations*

Anna Lubiw[1], Zuzana Masárová[2], and Uli Wagner[2]

[1]School of Computer Science, University of Waterloo , Waterloo, ON, Canada, N2L 3G1, alubiw@uwaterloo.ca
[2]IST Austria, Am Campus 1, 3400 Klosterneuburg, Austria, zuzana.masarova@ist.ac.at, uli@ist.ac.at

1 Introduction

Given a triangulation of a point set in the plane, a *flip* deletes an edge e whose removal leaves a convex quadrilateral, and replaces e by the opposite diagonal of the quadrilateral. It is well known that any triangulation of a point set can be reconfigured to any other triangulation by some sequence of flips. We explore this question in the setting where each edge of a triangulation has a label, and a flip transfers the label of the removed edge to the new edge. We characterize when one labelled triangulation of a point set can be reconfigured to another one via a sequence of flips. There is an obvious necessary condition: for each label l, if edge e has label l in the first triangulation and edge f has label l in the second triangulation, then there must be some sequence of flips that moves label l from e to f, ignoring all other labels. Bose, Lubiw, Pathak and Verdonschot [4] formulated the "*Orbit Conjecture*," which states that this necessary condition is also sufficient, i.e. that *all* labels can be simultaneously mapped to their destination if and only if *each* label individually can be mapped to its destination.

We prove this conjecture. Furthermore, we give a polynomial-time algorithm to find a sequence of flips to reconfigure one labelled triangulation to another, if such a sequence exists, and we prove an upper bound of $O(n^7)$ on the length of the flip sequence.

Our proof uses the topological result that straight-edge plane graphs on a planar point set form a simplicial complex that is homeomorphic to a high-dimensional ball (this follows from a result of Orden and Santos; we present here a different proof based on a shelling argument). The dual cell complex of this simplicial ball, which we call the *flip complex*, has the usual flip graph as its 1-skeleton. We use properties of the 2-skeleton of the flip complex.

We now fill in further details. Throughout, we fix a set P of n points in general position, and we identify triangulations with their edge sets (i.e., a triangulation of P is a maximal set T of pairwise non-crossing edges spanned by P). The result that any triangulation can be flipped to any other (see [3]) is captured succinctly by saying that the *flip graph* is connected, where the *flip graph* has a vertex for each triangulation of the given point set, and an edge when two triangulations differ by one flip. This connectivity result extends to *constrained triangulations* where some edges between points of P are fixed and not flippable (see [3]).

A *labelled triangulation* $\mathcal{T}$ of P is a pair (T, ℓ) where T is a triangulation of P and ℓ is a *labelling function* that maps the edges of T one-to-one onto the labels $1, 2, \ldots, t_P$. Here t_P is the number of edges in any triangulation of P. When we perform a flip operation on $\mathcal{T}$, the label of the removed edge is transferred to the new edge.

Edges e and f lie in the same *orbit* if we can attach label l to e in some triangulation and apply some sequence of flips to arrive at a triangulation in which edge f has label l. The orbits are exactly the connected components of a graph that Eppstein [5] called the *quadrilateral graph*—this graph has a vertex for every one of the possible $\binom{n}{2}$ edges formed by point set P, with e and f being adjacent if they cross and their four endpoints form a convex quadrilateral that is empty of other points. In particular, this implies that there is a polynomial-time algorithm to find the orbits.

The Orbit Conjecture of Bose et al. [4] can be expressed precisely using the terminology of "orbits". Our main result is to prove the Orbit Conjecture and strengthen it by providing a polynomial-time algorithm and a bound on the length of the flip sequence:

Theorem 1 (Orbit Theorem) *Given two edge-labelled triangulations $\mathcal{T}_1$ and $\mathcal{T}_2$ of a point set, there is a flip sequence that transforms one into the other if and only if for every label l, the edges of $\mathcal{T}_1$ and $\mathcal{T}_2$ having label l belong to the same orbit. Furthermore, there is a polynomial-time algorithm that tests whether the condition is satisfied, and if it is, computes a flip sequence of length $O(n^7)$ to transform $\mathcal{T}_1$ to $\mathcal{T}_2$.*

The crucial case is when the two triangulations have the same edge set but different label functions, i.e. we are given a permutation of the edge labels of a triangulation, and we seek a flip sequence to realize the permutation. Since every permutation is a composition of transpositions, we concentrate first on finding a flip sequence to transpose (or "swap") two labels.

One insight to be gained from previous work is that

*to appear, Symposium on Computational Geometry, 2017

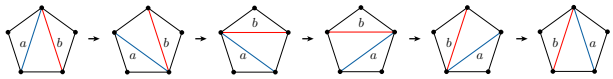


Figure 1: Five flips swap the edge labels (a and b) of two diagonals of a convex pentagon. In the flip graph these five flips form a 5-cycle.

empty convex pentagons in the point set seem to be crucial for swapping edge labels. Figure 1 shows how the edge labels of two diagonals of an empty convex pentagon can be swapped by a sequence of five flips.

We make this insight rigorous by showing that the only elementary operation that is needed for label permutation is to transpose two labels by moving them into an empty convex pentagon and swapping them there. More formally, given a labelled triangulation $\mathcal{T} = (\mathcal{T}, \ell)$, an *elementary swap* of edges e and f in T is a transposition of the labels of e and f that is accomplished as follows: perform a sequence, σ, of flips on $\mathcal{T}$ to get to a triangulation $\mathcal{T}'$ in which the labels $\ell(e)$ and $\ell(f)$ are attached to the two diagonals of an empty convex pentagon; then perform the 5-flip sequence, π, that transposes these two labels; then perform the sequence σ^{-1}. We say that the sequence $\sigma\pi\sigma^{-1}$ *realizes* the elementary swap. Observe that the effect of $\sigma\pi\sigma^{-1}$ on $\mathcal{T}$ is to transpose the labels of e and f while leaving all other labels unchanged. We will prove that an elementary swap can always be realized by a flip sequence of length $O(n^6)$, and furthermore, that such a sequence can be found in polynomial time.

One of our main results is the following, from which the Orbit Theorem can readily be derived:

Theorem 2 *In a labelled triangulation $\mathcal{T}$, two edges are in the same orbit if and only if there is an elementary swap between them.*

To prove Theorem 2, we use the following key result:

Theorem 3 (Elementary Swap Theorem) *Given a labelled triangulation $\mathcal{T}$, any permutation of the labels that can be realized by a sequence of flips can be realized by a sequence of elementary swaps.*

This theorem is proved using topological properties of the *flip complex*, whose 1-skeleton is the flip graph. A result of Orden and Santos [6] can be used to show that the flip complex has the topology of a high-dimensional ball[1]. We give an alternate proof of this. We use the 2-skeleton of the flip complex, and show that its 2-cells correspond to cycles in the flip graph of two types: quadrilaterals, which do not permute labels; and pentagons, which correspond precisely to the 5-cycles of flips shown in Figure 1. Then we prove the Elementary Swap Theorem by translating it into a result about decomposing closed walks in the flip graph into simpler *elementary walks*.

Although there is a rich literature on associahedra [8, 1] and on cell complexes associated with triangulations of point sets, we are not aware of any previous combinatorial results on triangulations that require topological proofs, as our proof of the Orbit Theorem seems to do.

[1] Technically speaking, the flip complex is homotopy equivalent to a ball.

2 Proof of the Orbit Theorem

In this section we prove the Orbit Theorem assuming the Elementary Swap Theorem (Theorem 3, proved in Section 3), and assuming the following two results on elementary swaps. We give combinatorial proofs of these lemmas in the longer version of the paper.

Lemma 4 *If there is an elementary swap between two edges in a triangulation $\mathcal{T}$ then there is a flip sequence of length $O(n^6)$ to realize the elementary swap. Furthermore, this sequence can be found in polynomial time.*

To prove Lemma 4 we look at paths in the *double quadrilateral graph* which has $O(n^4)$ vertices that correspond to pairs of non-crossing edges on point set P, and edges that correspond to flip sequences of length $O(n^2)$.

Lemma 5 *Let $\mathcal{T}$ be a labelled triangulation containing two edges e and f. If there is a sequence of elementary swaps on $\mathcal{T}$ that takes the label of edge e to edge f, then there is an elementary swap of e and f in $\mathcal{T}$.*

We prove the Orbit Theorem in stages, first Theorem 2, then the more general case of permuting edge labels in a triangulation, and finally the full result.

Proof. [Proof of Theorem 2] The "if" direction is clear, so we address the "only if" direction. Suppose that edges e and f are in the same orbit. Then there is a sequence of flips that changes the given edge-labelled triangulation $\mathcal{T} = (T, \ell)$ to an edge-labelled triangulation $\mathcal{T}' = (T', \ell')$ in which $\ell'(f) = \ell(e)$. We now apply the result that any constrained triangulation of a point set can be flipped to any other. Fix edge f and flip T' to T. Applying the same flip sequence to the labelled triangulation $\mathcal{T}'$ yields an edge-labelling of triangulation T in which edge f has the label $\ell(e)$. Thus we have a sequence of flips that permutes the labels of $\mathcal{T}$ and moves the label of e to f.

By the Elementary Swap Theorem (Theorem 3) there is a sequence of elementary swaps whose effect is to move the label of edge e to edge f. By Lemma 5 there is an elementary swap of e and f in $\mathcal{T}$. □

Theorem 6 (Edge Label Permutation Theorem) *Let T be a triangulation of a point set with two edge-labellings ℓ_1 and ℓ_2 such that for each label l, the edge with label l in ℓ_1 and the edge with label l in ℓ_2 are in the same orbit. Then there is a sequence of $O(n)$ elementary swaps to transform the first labelling to the second. Such a sequence can be realized via a sequence of $O(n^7)$ flips, which can be found in polynomial time.*

Proof. The idea is to effect the permutation as a sequence of swaps. If every edge has the same label in ℓ_1 and ℓ_2 we are done. So consider a label l that is attached to a different edge in ℓ_1 and in ℓ_2. Suppose $\ell_1(e) = l$ and $\ell_2(f) = l$, with $e \neq f$. By hypothesis, e and f are in the same orbit. By Theorem 2 there is an elementary swap of e and f in (T, ℓ_1) which results in a new labelling ℓ_1' that matches ℓ_2 in one more edge (namely the edge f) and still has the property that for every label l, the edge with label l in ℓ_1' and the edge with label l in ℓ_2 are in the same orbit. Thus we can continue this process until all edge labels match those of ℓ_2. In total we use $O(n)$ elementary swaps. These can be realized via a sequence of $O(n^7)$ flips by Lemma 4. Furthermore, the sequence can be found in polynomial time. □

We can now prove the Orbit Theorem.

Proof. [Proof of Theorem 1] The necessity of the condition is clear, and we can test it in polynomial time by finding all the orbits, so we address sufficiency.

The idea is to reconfigure $\mathcal{T}_1$ to have the same underlying unlabelled triangulation as $\mathcal{T}_2$ and then apply the previous theorem. The details are as follows. Let $\mathcal{T}_1 = (T_1, \ell_1)$ and $\mathcal{T}_2 = (T_2, \ell_2)$. There is a sequence σ of $O(n^2)$ flips to reconfigure the unlabelled triangulation T_1 to T_2, and σ can be found in polynomial time. Applying σ to the labelled triangulation $\mathcal{T}_1$ yields a labelled triangulation $\mathcal{T}_3 = (T_2, \ell_3)$. Note that for every label l, the edges of $\mathcal{T}_1$ and $\mathcal{T}_3$ having label l belong to the same orbit. This is because flips preserve orbits (by definition of orbits). Thus by Theorem 6 there is a flip sequence τ that reconfigures $\mathcal{T}_3$ to $\mathcal{T}_2$, and this flip sequence can be found in polynomial time and has length $O(n^7)$. The concatenation of the two flip sequences, $\sigma\tau$, reconfigures $\mathcal{T}_1$ to $\mathcal{T}_2$, has length $O(n^7)$, and can be found in polynomial time. □

There is a gap between our bound of $O(n^7)$ and the best known lower bound of $\Omega(n^3)$ on the length of a flip sequence to reconfigure one labelled triangulation to another (when this is possible) [4].

3 Proof of the Elementary Swap Theorem

As mentioned in the introduction, we use topological properties of the *flip complex*, whose 1-skeleton (i.e. vertices and edges) is the flip graph. In the full version of the paper we show that each 2-cell of the flip complex corresponds to a set of triangulations that differ on two edges. Specifically, we define an *elementary 4-cycle* and an *elementary 5-cycle* as in Figure 2.

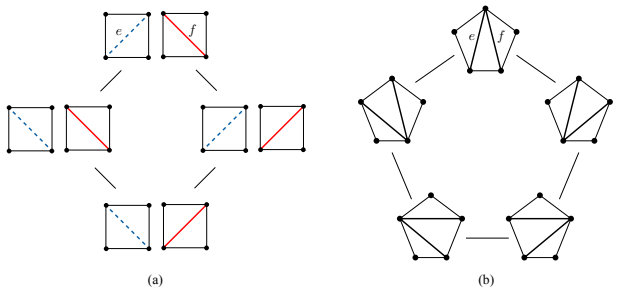


Figure 2: (a) Triangulations that differ in the diagonals of two internally disjoint quadrilaterals form an *elementary 4-cycle* in the flip graph. A walk around the cycle does not permute the labels (shown as red and blue). (b) Triangulations that differ in the diagonals of a convex pentagon form an *elementary 5-cycle* in the flip graph. This cycle permutes labels as shown in Figure 1.

The key to proving the Elementary Swap Theorem is the following topological result.

Theorem 7 *Let P be a set of n points in general position in the plane. There is a high-dimensional cell complex $\mathbb{X} = \mathbb{X}(P)$, which we call the* flip complex, *with the following properties:*

1. *The 1-skeleton of $\mathbb{X}$ is the flip graph of P;*
2. *The 2-cells of $\mathbb{X}$ correspond to the elementary 4-cycles and elementary 5-cycles of the flip graph;*
3. *$\mathbb{X}$ has the topology of (i.e., is homotopy equivalent to) a high-dimensional ball; therefore its* fundamental group, *$\pi_1(\mathbb{X})$, is trivial.*

Theorem 7 can be derived from a result of Orden and Santos [6]. In the long version of our paper we give an alternative proof of Theorem 7 that starts out by considering the simplicial complex $\mathbb{T} = \mathbb{T}(T)$ whose faces are the sets of pairwise non-crossing edges (line segments) spanned by P. This complex $\mathbb{T}$ is shown to be a *shellable simplicial ball* (by an argument based on constrained Delaunay triangulations and a theorem for shellable pseudomanifolds, see [2, Prop. 4.7.22]), and $\mathbb{X}$ is then constructed as the *dual complex* of $\mathbb{T}$.

To prove the Elementary Swap Theorem, we need to look at walks in the flip graph. Fix a triangulation T_0. An *elementary quadrilateral walk* is a closed walk of the form wzw^{-1}, where z is an elementary 4-cycle in the flip graph, and w is a walk from T_0 to some triangulation on z. An *elementary pentagonal walk* is

defined analogously, with z an elementary 5-cycle. It is straightforward to check the effect of these elementary walks on labellings:

Lemma 8 *Let (T_0, ℓ) be a labelled triangulation. An elementary quadrilateral walk does not permute the labels. An elementary pentagonal walk swaps the labels of two edges (e and f in Figure 2(b)) and leaves all other labels fixed; this corresponds exactly to the notion of an elementary swap introduced earlier.*

Another operation that does not affect the permutation of labels induced by a closed walk is the following. A *spur* ww^{-1} starting and ending at T is an arbitrary walk w starting at T, immediately followed by the *inverse walk*. If w_1 and w_2 are walks in the flip graph such that w_1 ends at a triangulation T and w_2 starts there, and if s is a spur at T, then we say that the walk $w_1 s w_2$ differs from $w_1 w_2$ by a *spur insertion* (and the inverse operation is called a *spur deletion*). In the long version we prove:

Lemma 9 *Two closed walks that differ only by a finite number of spur insertions and deletions yield the same permutation of edge labels.*

By Lemmas 8 and 9, the Elementary Swap Theorem directly reduces to the following, which we prove using Theorem 7:

Proposition 10 *Any closed walk in the flip graph is—up to a finite number of spur insertions and deletions—a composition of finitely many elementary walks.*

Proof. We use the well-known fact that in a cell complex with trivial fundamental group any two closed walks starting at the same vertex are related by a finite number of spur insertions, deletions and so-called 2-cell relations. The fundamental group of such a complex can be defined *combinatorially* in terms of closed walks in the 1-skeleton and this definition is equivalent to the usual topological definition in terms of continuous loops, see [7, Chap. 7] or [9, Chap. 4].

Specifically, we fix a *base triangulation* T_0, and, for every triangulation T, we fix a walk p_T from T_0 to T. Given two triangulations T_1, T_2 that differ by a flip, we form the closed walk w_{T_1,T_2} in the flip graph, called a *generating walk*, that goes from T_0 to T_1 along p_{T_1}, then flips to T_2, and then returns to T_0 along $p_{T_2}^{-1}$. It is easy to see that every closed walk starting and ending at T_0 can be written as a composition of generating walks.

We say that walks w and w' are 2-*cell related* if we can express them as $w = w_1 w_2$ and $w' = w_1 z w_2$, where z is a closed walk traversing the boundary of a 2-cell (an elementary cycle) exactly once in either orientation. A priori, this is not a symmetric relation, but $w_1 w_2$ and $w_1 z z^{-1} w_2$ differ only by the spur zz^{-1}. Also, notice the *precomposition property*: if w is precomposed with the elementary closed walk $(w_1 z w_1^{-1})$ then the result $w'' = (w_1 z w_1^{-1})w = w_1 z (w_1^{-1} w_1) w_2$ differs from w' only by the spur $(w_1^{-1} w_1)$.

Two walks in the flip graph are called equivalent if they differ by a finite number of spur insertion and/or deletions and by applying a finite number of 2-cell relations. It is not hard to check that this defines an equivalence relation, and the fundamental group $\pi_1(\mathbb{X})$ is given as the set of equivalence classes of closed walks starting and ending at T_0.

Triviality of the fundamental group translates into the fact that every closed walk starting and ending at T_0 is equivalent to the trivial walk. By the precomposition property, this means that, up to a finite number of spur insertions and deletions, every closed walk is a composition of finitely many elementary walks. □

References

[1] Gabriela Araujo-Pardo, Isabel Hubard, Deborah Oliveros, and Egon Schulte. Colorful associahedra and cyclohedra. *Journal of Combinatorial Theory, Series A*, 129:122–141, 2015.

[2] Anders Björner, Michel Las Vergnas, Bernd Sturmfels, Neil White, and Günter M. Ziegler. *Oriented matroids*, volume 46 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, second edition, 1999.

[3] Prosenjit Bose and Ferran Hurtado. Flips in planar graphs. *Computational Geometry Theory and Applications*, 42(1):60–80, 2009.

[4] Prosenjit Bose, Anna Lubiw, Vinayak Pathak, and Sander Verdonschot. Flipping edge-labelled triangulations. *arXiv preprint arXiv:1310.1166*, 2013. To appear in *Computational Geometry*.

[5] David Eppstein. Happy endings for flip graphs. *Journal of Computational Geometry*, 1(1):3–28, 2010.

[6] David Orden and Francisco Santos. The polytope of non-crossing graphs on a planar point set. *Discrete Comput. Geom.*, 33(2):275–305, 2005.

[7] Herbert Seifert and William Threlfall. *A Textbook of Topology*, volume 89 of *Pure and Applied Mathematics*. Academic Press, 1980.

[8] Daniel D. Sleator, Robert E. Tarjan, and William P. Thurston. Rotation distance, triangulations, and hyperbolic geometry. *Journal of the American Mathematical Society*, 1(3):647–681, 1988.

[9] John Stillwell. *Classical topology and combinatorial group theory*, volume 72 of *Graduate Texts in Mathematics*. Springer-Verlag, second edition, 1993.

Minimum Perimeter-Sum Partitions in the Plane*

Mikkel Abrahamsen†, Mark de Berg‡, Kevin Buchin‡, Mehran Mehr‡, Ali D. Mehrabi‡

Abstract

Let P be a set of n points in the plane. We consider the problem of partitioning P into two subsets P_1 and P_2 such that the sum of the perimeters of $\text{CH}(P_1)$ and $\text{CH}(P_2)$ is minimized, where $\text{CH}(P_i)$ denotes the convex hull of P_i. The problem was first studied by Mitchell and Wynters in 1991 who gave an $O(n^2)$ time algorithm. Despite considerable progress on related problems, no subquadratic time algorithm for this problem was found so far. We present an algorithm solving the problem in $O(n \log^4 n)$ time.

1 Introduction

A natural class of clustering problems is to divide a set P of points in the plane into two clusters P_1 and P_2 so as to minimize a cost function expressing the size of the convex hulls $\text{CH}(P_1)$ and $\text{CH}(P_2)$. For instance, $\text{CH}(P_i)$ can be defined as the area of $\text{CH}(P_i)$ or as the perimeter $\text{per}(P_i)$ of $\text{CH}(P_i)$. (The perimeter of $\text{CH}(P_i)$ is the length of the boundary $\partial\,\text{CH}(P_i)$.) Such clustering problems were already studied in 1991 by Mitchell and Wynters [3]. They studied four problem variants: minimize the sum of the perimeters, the maximum of the perimeters, the sum of the areas, or the maximum of the areas. In three of the four variants the convex hulls $\text{CH}(P_1)$ and $\text{CH}(P_2)$ in an optimal solution may intersect—only in the *minimum perimeter-sum problem* the optimal bipartition is guaranteed to be a so-called *line partition*, that is, a solution with disjoint convex hulls. For each of the four variants they gave an $O(n^3)$ algorithm that uses $O(n)$ space and for all except the minimum-maximum area problem, they also gave an $O(n^2)$ algorithm that uses $O(n^2)$ space; their algorithms only consider line partitions (which in the case of the minimum perimeter-sum problem implies an optimal bipartition). Mitchell and Wynters mentioned the improvement of the space requirement of the quadratic-time algorithm as an open problem, and they stated the existence of a subquadratic algorithm for any of the four variants as the most prominent open problem.

Rokne *et al.* [4] made progress on the first question, by presenting an $O(n^2 \log n)$ algorithm that uses only $O(n)$ space for the line-partition version of each of the four problems. The main question is still open: is it possible to obtain a subquadratic algorithm for any of the four bipartition problems based on convex-hull size?

Our contribution. We answer the question above affirmatively by presenting a subquadratic algorithm for the minimum perimeter-sum bipartition problem in the plane.

As mentioned, an optimal solution (P_1, P_2) to the minimum perimeter-sum bipartition problem must be a line partition. A straightforward algorithm would generate all $\Theta(n^2)$ line partitions and compute the value $\text{per}(P_1) + \text{per}(P_2)$ for each of them. If the latter is done from scratch for each partition, the resulting algorithm runs in $O(n^3 \log n)$ time. The algorithms by Mitchell and Wynters [3] and Rokne *et al.* [4] improve on this by using that the different line bipartitions can be generated in an ordered way, such that subsequent line partitions differ in at most one point. Thus the convex hulls do not have to be recomputed from scratch, but they can be obtained by updating the convex hulls of the previous bipartition. To obtain a subquadratic algorithm a fundamentally new approach is necessary: we need a strategy that generates a subquadratic number of candidate partitions, instead considering all line partitions. We achieve this as follows.

We start by presenting a theorem stating that an optimal bipartition (P_1, P_2) has the following property: there is a set of $O(1)$ canonical orientations such that P_1 can be separated from P_2 by a line with a canonical orientation, or the distance between $\text{CH}(P_1)$ and $\text{CH}(P_2)$ is $\Omega(\min(\text{per}(P_1), \text{per}(P_2))$. There are only $O(1)$ bipartitions of the former type, and finding the best among them is relatively easy. The bipartitions of the second type are much more challenging. We show how to employ a compressed quadtree to generate a collection of $O(n)$ canonical 5-gons—intersections of axis-parallel rectangles and canonical halfplanes—such that the smaller of $\text{CH}(P_1)$ and $\text{CH}(P_2)$ (in a bipartition of the second type) is contained in one of the 5-gons.

It then remains to find the best among the bipartitions of the second type. Even though the number

*MA is partly supported by Mikkel Thorup's Advanced Grant from the Danish Council for Independent Research under the Sapere Aude research career programme. MdB, KB, MM, and AM are supported by the Netherlands' Organisation for Scientific Research (NWO) under project no. 024.002.003, 612.001.207, 022.005025, and 612.001.118 respectively.

†University of Copenhagen, mia@di.ku.dk.

‡TU Eindhoven, mdberg@win.tue.nl, k.a.buchin@tue.nl, m.mehr@tue.nl, amehrabi@win.tue.nl.

of such bipartitions is linear, we cannot afford to compute their perimeters from scratch. We therefore design a data structure to quickly compute $\mathrm{per}(P \cap Q)$, where Q is a query canonical 5-gon. Brass *et al.* [1] presented such a data structure for the case where Q is an axis-parallel rectangle. Their structure uses $O(n \log^2 n)$ space and has $O(\log^5 n)$ query time; it can be extended to handle canonical 5-gons as queries, at the cost of increasing the space usage to $O(n \log^3 n)$ and the query time to $O(\log^7 n)$. Our data structure improves upon this: it has $O(\log^4 n)$ query time for canonical 5-gons (and $O(\log^3 n)$ for rectangles) while using the same amount of space. Using this data structure to find the best bipartition of the second type we obtain our main result: an exact algorithm for the minimum perimeter-sum bipartition problem that runs in $O(n \log^4 n)$ time.

2 The algorithm

Let P be the set of n points in the plane for which we want to solve the minimum-perimeter-sum partition problem. An optimal partition (P_1, P_2) of P has the following two basic properties: P_1 and P_2 are non-empty, and the convex hulls $\mathrm{CH}(P_1)$ and $\mathrm{CH}(P_2)$ are disjoint. In the remainder, whenever we talk about a partition of P, we refer to a partition with these two properties.

Consider a partition (P_1, P_2) of P. Define $\mathcal{P}_1 := \mathrm{CH}(P_1)$ and $\mathcal{P}_2 := \mathrm{CH}(P_2)$ to be the convex hulls of P_1 and P_2, respectively, and let ℓ_1 and ℓ_2 be the two inner common tangents of $\mathcal{P}_1$ and $\mathcal{P}_2$, see Figure 1. The lines ℓ_1 and ℓ_2 define four wedges: one containing P_1, one containing P_2, and two empty wedges. We call the opening angle β of the empty wedges the *separation angle* of P_1 and P_2. Furthermore, we call the distance between $\mathcal{P}_1$ and $\mathcal{P}_2$ the *separation distance* of P_1 and P_2. Our algorithm relies on the following separation property of any optimal bipartition.

Theorem 1 *Let P be a set of n points in the plane, and let (P_1, P_2) be a partition of P that minimizes $\mathrm{per}(P_1) + \mathrm{per}(P_2)$. Then the separation angle of P_1 and P_2 is at least $\pi/6$ or the separation distance is at least $c_{\mathrm{sep}} \cdot \min(\mathrm{per}(P_1), \mathrm{per}(P_2))$, where $c_{\mathrm{sep}} := 1/250$.*

The proof of this theorem is rather delicate and can be found in the full version. Below we sketch the main ideas. Let (P_1, P_2) be a partition of P that minimizes $\mathrm{per}(P_1) + \mathrm{per}(P_2)$. Let ℓ_3 and ℓ_4 be the outer common tangents of $\mathcal{P}_1$ and $\mathcal{P}_2$. We define α to be the angle between ℓ_3 and ℓ_4. More precisely, if ℓ_3 and ℓ_4 are parallel we define $\alpha := 0$, otherwise we define α as the opening angle of the wedge defined by ℓ_3 and ℓ_4 containing $\mathcal{P}_1$ and $\mathcal{P}_2$.

Suppose that the separation distance and the separation angle β are both relatively small. Then the region

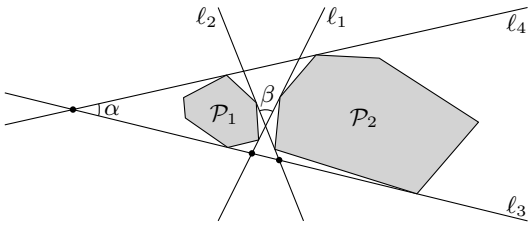


Figure 1: The angles α and β.

A in between $\mathcal{P}_1$ and $\mathcal{P}_2$ and bounded from the bottom by ℓ_3 and from the top by ℓ_4 is relatively narrow. But then the left and right parts of ∂A (which are contained in $\partial \mathcal{P}_1$ and $\partial \mathcal{P}_2$) would be longer than the bottom and top parts of ∂A (which are contained in ℓ_3 and ℓ_4), thus contradicting that (P_1, P_2) is an optimal partition. To make this idea precise, we first prove that if the separation angle β is small, then the angle α between ℓ_3 and ℓ_4 must be large. Second, we show that there is a value $f(\alpha)$ such that the distance between $\mathcal{P}_1$ and $\mathcal{P}_2$ is at least $f(\alpha) \cdot \min(\mathrm{per}(P_1), \mathrm{per}(P_2))$. Finally we argue that this implies that if the separation angle is smaller than $\pi/6$, then (to avoid the contradiction mentioned above) the separation distance must be relatively large.

Theorem 1 suggests to distinguish two cases when computing an optimal partition: the case where the separation angle is large (namely at least $\pi/6$) and the case where the separation distance is large (namely at least $c_{\mathrm{sep}} \cdot \min(\mathrm{per}(P_1), \mathrm{per}(P_2))$). As we will see, the first case can be handled in $O(n \log n)$ time and the second case in $O(n \log^4 n)$ time, leading to the following theorem.

Theorem 2 *Let P be a set of n points in the plane. Then we can compute a partition (P_1, P_2) of P that minimizes $\mathrm{per}(P_1) + \mathrm{per}(P_2)$ in $O(n \log^4 n)$ time using $O(n \log^3 n)$ space.*

The rest of this section is dedicated to the proof of Theorem 2. To find the best partition when the separation angle is at least $\pi/6$, we observe that in this case there is a separating line whose orientation is $j \cdot \pi/7$ for some $0 \leqslant j < 7$. For each of these orientations we can scan over the points with a line ℓ of the given orientation, and maintain the perimeters of the convex hulls on both sides. This takes $O(n \log n)$ time in total.

Next we show how to compute the best partition with large separation distance. We assume without loss of generality that $\mathrm{per}(P_2) \leqslant \mathrm{per}(P_1)$. It will be convenient to treat the case where P_2 is a singleton separately.

Lemma 3 *The point $p \in P$ minimizing $\mathrm{per}(P \setminus \{p\})$ can be computed in $O(n \log n)$ time.*

It remains to compute the best partition (P_1, P_2) with $\text{per}(P_2) \leqslant \text{per}(P_1)$ whose separation distance is at least $c_{\text{sep}} \cdot \text{per}(P_2)$ and where P_2 is not a singleton. Let (P_1^*, P_2^*) denote this partition. Define the *size* of a square (whenever we speak of squares, we always mean axis-parallel squares) σ to be its edge length. A square σ is a *good square* if (i) $P_2^* \subset \sigma$, and (ii) $\text{size}(\sigma) \leqslant c^* \cdot \text{per}(P_2^*)$, where $c^* := 18$. Our algorithm globally works as follows.

1. Compute a set S of $O(n)$ squares such that S contains a good square.
2. For each square $\sigma \in S$, construct a set H_σ of $O(1)$ halfplanes such that the following holds: if $\sigma \in S$ is a good square then there is a halfplane $h \in H_\sigma$ such that $P_2^* = P(\sigma \cap h)$, where $P(\sigma \cap h) := P \cap (\sigma \cap h)$.
3. For each pair (σ, h) with $\sigma \in S$ and $h \in H_\sigma$, compute $\text{per}(P \setminus P(\sigma \cap h)) + \text{per}(P(\sigma \cap h))$, and report the partition $(P \setminus P(\sigma \cap h), P(\sigma \cap h))$ that gives the smallest sum.

Step 1: Finding a good square. To find a set S that contains a good square, we first construct a set S_{base} of so-called *base squares*. The set S will then be obtained by expanding the base squares appropriately.

We define a base square σ to be *good* if (i) σ contains at least one point from P_2^*, and (ii) $c_1 \cdot \text{diam}(P_2^*) \leqslant \text{size}(\sigma) \leqslant c_2 \cdot \text{diam}(P_2^*)$, where $c_1 := 1/4$ and $c_2 := 4$ and $\text{diam}(P_2^*)$ denotes the diameter of P_2^*. Note that $2 \cdot \text{diam}(P_2^*) \leqslant \text{per}(P_2^*) \leqslant 4 \cdot \text{diam}(P_2^*)$. For a square σ, define $\overline{\sigma}$ to be the square with the same center as σ and whose size is $(1 + 2/c_1) \cdot \text{size}(\sigma)$.

Lemma 4 *If σ is a good base square then $\overline{\sigma}$ is a good square.*

To obtain S it thus suffices to construct a set S_{base} that contains a good base square. To this end we first build a compressed quadtree for P. For completeness we briefly review the definition of compressed quadtrees; see also Fig. 2 (left).

Assume without loss of generality that P lies in the interior of the unit square $U := [0, 1]^2$. Define a *canonical square* to be any square that can be obtained by subdividing U recursively into quadrants. A *compressed quadtree* [2] for P is a hierarchical subdivision of U, defined as follows. In a generic step of the recursive process we are given a canonical square σ and the set $P(\sigma) := P \cap \sigma$ of points inside σ. (Initially $\sigma = U$ and $P(\sigma) = P$.)

- If $|P(\sigma)| \leqslant 1$ then the recursive process stops and σ is a square in the final subdivision.
- Otherwise there are two cases. Consider the four quadrants of σ. The first case is that at least two of these quadrants contain points from $P(\sigma)$. (We consider the quadrants to be closed on the left and bottom side, and open on the right and top side, so a point is contained in a unique quadrant.) In this case we partition σ into its four quadrants—we call this a *quadtree split*—and recurse on each quadrant. The second case is that all points from $P(\sigma)$ lie inside the same quadrant. In this case we compute the smallest canonical square, σ', that contains $P(\sigma)$ and we partition σ into two regions: the square σ' and the so-called *donut region* $\sigma \setminus \sigma'$. We call this a *shrinking step*. After a shrinking step we only recurse on the square σ', not on the donut region.

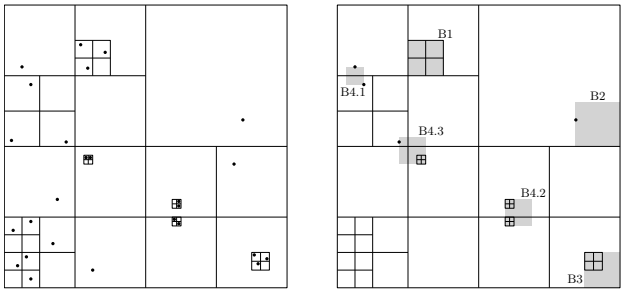


Figure 2: A compressed quadtree and some of the base squares generated from it. In the right figure, only the points are shown that are relevant for the shown base squares.

A compressed quadtree for a set of n points can be computed in $O(n \log n)$ time in the appropriate model of computation [2]. The idea is now as follows. Let $p, p' \in P_2^*$ be a pair of points defining $\text{diam}(P_2^*)$. The compressed quadtree hopefully allows us to zoom in until we have a square in the compressed quadtree that contains p or p' and whose size is roughly equal to $|pp'|$. Such a square will be then a good base square. Unfortunately this does not always work since p and p' can be separated too early. We therefore have to proceed more carefully: we need to add five types of base squares to S_{base}, as explained next and illustrated in Fig. 2 (right).

(B1) Any square σ that is generated during the recursive construction—note that this not only refers to squares in the final subdivision—is put into S_{base}.

(B2) For each point $p \in P$ we add a square σ_p to S_{base}, as follows. Let σ be the square of the final subdivision that contains p. Then σ_p is a smallest square that contains p and that shares a corner with σ.

(B3) For each square σ that results from a shrinking step we add an extra square σ' to S_{base}, where σ'

is the smallest square that contains σ and that shares a corner with the parent square of σ.

(B4) For any two regions in the final subdivision that touch each other—we also consider two regions to touch if they only share a vertex—we add at most one square to S_{base}, as follows. If one of the regions is an empty square, we do not add anything for this pair. Otherwise we have three cases.

- **(B4.1)** If both regions are non-empty squares containing points p and p', respectively, then we add a smallest enclosing square for the pair of points p, p' to S_{base}.
- **(B4.2)** If both regions are donut regions, say $\sigma_1 \setminus \sigma_1'$ and $\sigma_2 \setminus \sigma_2'$, then we add a smallest enclosing square for the pair σ_1', σ_2' to S_{base}.
- **(B4.3)** If one region is a non-empty square containing a point p and the other is a donut region $\sigma \setminus \sigma'$, then we add a smallest enclosing square for the pair p, σ' to S_{base}.

Lemma 5 *The set S_{base} has size $O(n)$ and contains a good base square. Furthermore, S_{base} can be computed in $O(n \log n)$ time.*

Step 2: Generating halfplanes. Consider a good square $\sigma \in S$. Let Q_σ be a set of $4 \cdot c^*/c_{\text{sep}} + 1 = 18001$ points placed equidistantly around the boundary of σ. Note that the distance between two neighbouring points in Q_σ is less than $c_{\text{sep}}/c^* \cdot \text{size}(\sigma)$. For each pair q_1, q_2 of points in Q_σ, add to H_σ the two halfplanes defined by the line through q_1 and q_2.

Lemma 6 *For any good square $\sigma \in S$, there is a halfplane $h \in H_\sigma$ such that $P_2^* = P(\sigma \cap h)$.*

Step 3: Evaluating candidate solutions. In this step we need to compute for each pair (σ, h) with $\sigma \in S$ and $h \in H_\sigma$, the value $\text{per}(P \setminus P(\sigma \cap h)) + \text{per}(P(\sigma \cap h))$. We do this by preprocessing P into a data structure that allows us to quickly compute $\text{per}(P \setminus P(\sigma \cap h))$ and $\text{per}(P(\sigma \cap h))$ for a given pair (σ, h). Recall that the bounding lines of the halfplanes h we must process have $O(1)$ different orientations. We construct a separate data structure for each orientation.

Consider a fixed orientation ϕ. We build a data structure $\mathcal{D}_\phi$ for range searching on P with ranges of the form $\sigma \cap h$, where σ is a square and h is halfplane whose bounding line has orientation ϕ. Since the edges of σ are axis-parallel and the bounding line of the halfplanes h have a fixed orientation, we can use a standard three-level range tree for this. Constructing this tree takes $O(n \log^2 n)$ time and the tree has $O(n \log^2 n)$ nodes.

Each node ν of the third-level trees in $\mathcal{D}_\phi$ is associated with a *canonical subset* $P(\nu)$, which contains the points stored in the subtree rooted at ν. We preprocess each canonical subset $P(\nu)$ as follows. First we compute the convex hull $\text{CH}(P(\nu))$. Let $v_1, \ldots, v_k$ denote the convex-hull vertices in counterclockwise order. We store these vertices in order in an array, and we store for each vertex v_i the value $\text{length}(\partial P(v_1, v_i))$, that is, the length of the part of $\partial\,\text{CH}(P(\nu))$ from v_1 to v_i in counterclockwise order. Note that the convex hull $\text{CH}(P(\nu))$ can be computed in $O(|P(\nu)|)$ from the convex hulls at the two children of ν. Hence, the convex hulls $\text{CH}(P(\nu))$ (and the values $\text{length}(\partial P(v_1, v_i))$) can be computed in $\sum_{\nu \in \mathcal{D}_\phi} O(|P(\nu)|) = O(n \log^3 n)$ time in total, in a bottom-up manner.

Now suppose we want to compute $\text{per}(P(\sigma \cap h))$, where the orientation of the bounding line of h is ϕ. We perform a range query in $\mathcal{D}_\phi$ to find a set $N(\sigma \cap h)$ of $O(\log^3 n)$ nodes such that $P(\sigma \cap h)$ is equal to the union of the canonical subsets of the nodes in $N(\sigma \cap h)$. Standard range-tree properties guarantee that the convex hulls $\text{CH}(P(\nu))$ and $\text{CH}(P(\mu))$ of any two nodes $\nu, \mu \in N(\sigma \cap h)$ are disjoint. Note that $\text{CH}(P(\sigma \cap h))$ is equal to the convex hull of the set of convex hulls $\text{CH}(P(\nu))$ with $\nu \in N(\sigma \cap h)$. A generalization of Andrew's version of Graham's scan for disjoint convex polygons instead of points now enables us to compute $\text{per}(P(\sigma \cap h))$ in $O(\log^4 n)$ time.

Observe that $P \setminus P(\sigma \cap h)$ can also be expressed as the union of $O(\log^3 n)$ canonical subsets with disjoint convex hulls, since $\mathbb{R}^2 \setminus (\sigma \cap h)$ is the disjoint union of $O(1)$ ranges of the right type. Hence, we can compute $\text{per}(P \setminus P(\sigma \cap h))$ in $O(\log^4 n)$ time. We thus obtain the following result, which finishes the proof of Theorem 2.

Lemma 7 *Step 3 can be performed in $O(n \log^4 n)$ time and using $O(n \log^3 n)$ space.*

References

[1] P. Brass, C. Knauer, C.-S. Shin, M. Smid, and I. Vigan. Range-aggregate queries for geometric extent problems. In *Proc. 19th CATS*, pages 3–10, 2013.

[2] S. Har-Peled. *Geometric approximation algorithms*. Mathematical surveys and monographs, Vol. 173. American Mathematical Society, 2011.

[3] J.S.B. Mitchell and E.L. Wynters. Finding optimal bipartitions of points and polygons. In *Proc. 2nd WADS*, LNCS 519, pages 202–213, 1991.

[4] J. Rokne, S. Wang, and X. Wu. Optimal bipartitions of point sets. In *Proc. 4th CCCG*, pages 11–16, 1992.

Searching edges in the overlap of two plane graphs

John Iacono* Elena Khramtcova† Stefan Langerman†

Abstract

Consider a pair R, B of plane straight-line graphs, whose edges are colored red and blue, respectively; let n be the total number of edges in R and B. We present a $O(n \log n)$-time $O(n)$-space technique to preprocess R, B that enables efficient searches for red-blue intersections along the red edges. Our technique has a number of applications to diverse geometric problems.

1 Introduction

Many geometric algorithms have subroutines that involve investigating intersections between two plane graphs, often assumed being colored red and blue respectively. Such subroutines differ in the questions that are asked about the red-blue intersections. Often these questions are to report all red-blue intersections or to count them. Reporting and counting the intersections can be carried out in $O(n)$ space and respectively $O(n \log n + k)$ and $O(n \log n)$ time, where n is the total complexity of both graphs, and k is the size of the output [2, 12]. Note that k may be $\Omega(n^2)$.

Here we consider the situation, where one wants to *search* the red-blue intersections, still avoiding to compute all of them. Problems of this type appear as building blocks in diverse algorithms, including distance measurement between polyhedral terrains [2], motion planning [9], construction of various generalized Voronoi diagrams [6, 4, 3]. Therefore solving such problems efficiently is of high importance.

Often the situation is as follows: each red edge contains at most one sought red-blue intersection, and there is an oracle that, given a red-blue intersection p, can quickly determine whether the sought intersection lies to the right or to the left of p along the same red edge. A particular case, when the red graph is a unique edge, appeared as *segment query* in [3], or as *find-change* query in [5]. If the blue graph is a tree, it can in $O(n \log n)$ time be preprocessed using the *centroid decomposition* [3]. Centroid decomposition supports segment (or find-change) queries for arbitrary line segments, requiring only $O(\log n)$ queries to the oracle [3]. If the blue graph is not a tree, then in $O(n \log n)$ time it can be preprocessed for point location, and a nested point location along the red edge is performed, which requires $O(\log^2 n)$ queries to the oracle [4, 5]. For two general plane straight-line graphs (where the red graph is not necessarily one edge) the problem is called *batched red-blue intersection problem*. It was formulated in Dehne et al. [6], and solved in $O(n \log^3 n)$ time and $O(n \log^2 n)$ space [6] using *hereditary segment trees* [2]. However, this is optimal in neither time nor space.

We present a data structure that provides a clear interface for efficient searches for red-blue intersections along a red edge. It can be used to improve the above result [6], which includes an improvement on segment (or find-change) queries in plane straight-line graphs. Our data structure can also handle more general search problems, e.g., a setting when one red edge may have more than one sought red-blue intersection on it.

1.1 Our result

Let R, B be a pair of plane straight-line graphs, and let n be the total complexity of both R and B. We address the following problem.

Problem 1.1 (RB-Preprocessing problem) *Given graphs R, B, construct a data structure that for each edge e of R stores implicitly the intersections between e and the edges of B sorted according to the order, in which these intersections appear along e. The data structure should support navigation in a perfectly balanced binary search tree T_e built on the sorted sequence of intersections along e:*
(i) Return the root of T_e;
(ii) Given a non-root node of T_e, return the parent of this node;
(iii) Given a non-leaf node of T_e, return the left (or, respectively, the right) child of this node.

We solve the RB-Preprocessing problem, such that the preprocessing takes $O(n \log n)$ time and $O(n)$ space and the above queries are supported in $O(1)$ time each.

The resulting data structure allows for fast searches for *interesting* intersections between the edges of R and the ones of B. We note that the notion of *interesting* is external to the data structure: It is not known at the

*Dept. of Computer Science and Engineering, New York University, USA. Research partially completed while J. I. was on sabbatical at the Algorithms Research Group of the Computer Science Dept. at ULB with support from a Fulbright Research Fellowship, F.R.S.-FNRS., and NSF grants CNS-1229185, CCF-1319648, and CCF-1533564.

†Computer Science Dept., Université Libre de Bruxelles, Belgium. E. K. was supported by F.R.S.-FNRS, and by the SNF grant P2TIP2-168563 of the Early PostDoc Mobility program; S. L. is directeur de recherches du F.R.S.-FNRS.

time of preprocessing, but rather guides the searches on the data structure after it is built. In particular, for the input graphs R and B, the data structure is always the same, while interesting intersections can be defined in several ways, which of course implies that the searches may have different outputs.

Our preprocessing technique can be applied to a number of geometric problems. We provide a list of applications, which is not exhaustive. For each application, we show how to reduce the initial problem to searching for interesting red-blue intersections, and how to navigate the searches, that is, how to decide, which subtree(s) of the current node of the (implicit) tree to search. As a result, by using our technique we are able to make the following contributions:

1. *The* batched red-blue search problem *[6] for a pair of segment sets can be solved in $O(n)$ space and $O(n \log n)$ queries to the oracle*, where n is the total number of segments in both sets. The problem is as follows. Given are: (1) two sets of line segments in the plane (colored red and blue, respectively), where no two segments in the same set intersect; (2) an oracle that, given an intersection point p of a red segment r with a blue segment b, determines to which side of r with respect to p the *interesting red-blue intersection* lies. Find all interesting red-blue intersections. (Each r has at most one such intersection). Our solution improves on the one by Dehne et al. [6] requiring $O(n \log^2 n)$ space and $O(n \log^3 n)$ queries to the oracle.

2. *The maximum vertical distance between a pair of polyhedral terrains, one of which is convex, can be computed in $O(n \log n)$ time and $O(n)$ space.* Previously, the *minimum* vertical distance between a pair of non-intersecting polyhedral terrains was considered. It was shown how to find it in $O(n^{4/3+\epsilon})$ time and space for a pair of general polyhedral terrains [2], in $O(n \log n)$ time for one convex and one general terrain [14], and in $O(n)$ time for two convex terrains [14]. Our technique yields a solution for the second case within the same time bound as in [14]. The *maximum* distance for non-intersecting polyhedra can be found as in [2, 14], but for intersecting polyhedra, it is different from the minimum distance: finding the former is still interesting, while the latter is trivially zero.

3. *The Hausdorff Voronoi diagram of a family of point clusters in the plane can be constructed in $O((n+m)\log^3 n)$ time*, where m is the total number of *crossings* of the clusters. Parameter m can be $\Theta(n^2)$, but is small in practice [13]. There is a deterministic algorithm to compute the diagram in $O(n^2)$ time [8]. All other known deterministic algorithms (see [13] and references therein) have a running time that depends on parameters of the input that cannot be bounded by a function of m. Each of them may take $\Omega(n^2)$ time even if clusters are non-crossing. There is a recent randomized algorithm with expected time complexity $O((m+n\log n)\log n))$ [11]. Thus the algorithm we propose here is the best deterministic algorithm for the cluster families with small number of crossings. The time complexity of our algorithm is subquadratic in n and m, and depends only on them, as opposed to any previous deterministic algorithm.

4. *The farthest-color Voronoi diagram of a family of n point clusters, where each cluster is the four corners of am axis-aligned rectangle, and all rectangles are pairwise disjoint, can be constructed in $O(n \log^2 n)$ time and $O(n)$ space.* Previous results on the topic are as follows. For a set of arbitrary point clusters, the diagram may have complexity $\Theta(n^2)$ and can be constructed in $O(n^2)$ time and space [1, 8], where n is the total number of points in all clusters. When clusters are pairs of endpoints of n parallel line segments, the diagram has $O(n)$ complexity and can be constructed in $O(n \log n)$ time and $O(n)$ space [5]. Here we broaden the class of inputs, for which the diagram can be constructed in subquadratic time.

5. *Stabbing circle problem for line segments in the plane can be solved in time $O(\mathcal{T}_{\mathsf{HVD}(S)} + \mathcal{T}_{\mathsf{FCVD}(S)} + (|\mathsf{HVD}(S)| + |\mathsf{FCVD}(S)| + m)\log n)$*, where $|\mathsf{HVD}(S)|$ and $|\mathsf{FCVD}(S)|$ denote respectively the complexity of the Hausdorff and the farthest-color Voronoi diagram of the pairs of endpoints of segments in S, $\mathcal{T}_{\mathsf{HVD}(S)}$ and $\mathcal{T}_{\mathsf{FCVD}(S)}$ denote the time to compute these diagrams, and m is a parameter reflecting the number of "bad" pairs of segments in S. If all segments in S are parallel to each other, the stabbing circle problem can be solved in optimal $O(n \log n)$ time and $O(n)$ space. This is an improvement over the recent $O(\mathcal{T}_{\mathsf{HVD}(S)} + \mathcal{T}_{\mathsf{FCVD}(S)} + (|\mathsf{HVD}(S)| + |\mathsf{FCVD}(S)| + m)\log^2 n)$ time technique for general segments, which yielded an $O(n \log^2 n)$ time algorithm for parallel segments [5].

In this short abstract we describe solely the technique, not the applications. For the full paper, see [10].

2 The technique to preprocess a pair of graphs

Below we treat R and B as two sets of line segments in the plane, colored respectively red and blue. No two segments of the same color intersect (although they may share an endpoint). We assume that no two distinct endpoints have the same x coordinate.

Our technique consists of three phases (see Sections 2.1, 2.2, and 2.3): First, we invoke an algorithm that finds the intersections between the red and the blue segments. Next, we build a linearized *life table* for the red segments. Finally, we sweep the life table with a line, which provides us the resulting data structure.

2.1 Finding red-blue intersections

The red-blue intersections in R, B can be counted or reported in $O(n)$ space and, respectively, $O(n \log n)$ or $O(k + n \log n)$ time, where k is the size of the output,

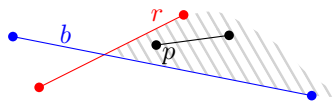


Figure 1: Line segments r and b, the closed right wedge formed by them, and the witness p of their intersection

as was shown by Mantler and Snoyeink [12]. Their algorithm processes the red-blue intersections in batches called *bundle-bundle* intersections. In $O(n \log n)$ time and $O(n)$ space it can (implicitly) discover all the intersections, without reporting every one of them individually. This is useful for our technique, which invokes the Mantler-Snoyeink algorithm in its first phase. Below we summarize the algorithm.

We first define (i) the *witness* of a (bichromatic) segment intersection, (ii) a *pseudoline at time* i, and (iii) a (monochromatic) *bundle* of segments at time i.

Given a red segment r that intersects a blue segment b, the *witness* of their intersection is the leftmost of the endpoints of segments in S that are contained in the closed right wedge formed by r and b. This wedge is the intersection of two closed right halfplanes: the one bounded by the line through r, and the one bounded by the line through b, see the shaded area in Figure 1. For each red-blue intersection there exists a witness.

Let $p_1, p_2, \ldots, p_{n'}$ be the sequence of distinct endpoints of the segments in R and B, ordered by increasing x coordinate, see gray numbers in Figure 2.

Lemma 1 *For each $i, 1 \leq i \leq n'$ there is a y-monotone curve ℓ_i that passes through point p_i, and subdivides the plane in two open regions (the left and the right one), such that all the points $p_j, j < i$, and all the red-blue intersections witnessed by the points $p_j, j \leq i$ are contained in the left region, and all the points $p_k, k > i$ together with the intersections witnessed by them are contained in the right region, and ℓ_i intersects each segment in R or in B at most once.*

We call such curve ℓ_i a *pseudoline at time* i. Figure 2 shows a pseudoline at time 7, i.e., ℓ_7, in dashed black lines. Note that ℓ_7 cannot be replaced by a vertical line as it must pass through the point 7, and to the left of the intersection point of segments r_4 and b_4, and the latter point lies to the left of the former one.

A *blue bundle* at time i is a maximal contiguous sequence of blue segments that intersect the pseudoline ℓ_i. A *red bundle* is defined analogously. Figure 2 lists all the blue bundles B_i, $1 \leq i \leq 10$, each with its edge sequence and the time at which it is first encountered.

The algorithm is a topological sweep by a pseudoline, where the only events are the endpoints $p_1, \ldots, p_{n'}$ of the segments in R and B. The sweepline at each moment i is a pseudoline ℓ_i, see Lemma 1. The sweepline status structure maintains all the red and blue bundles that intersect the current sweepline. It consists of (1) a balanced binary tree for each bundle, supporting insertion, deletion of segments, a query for the topmost and the bottommost segment in the bundle, and split and merge operations. (2) a list for all the bundles intersecting the sweepline, supporting insertion, deletion of the bundles, and sequential search, and (3) two balanced binary trees storing respectively all the red and all the blue bundles, and supporting splitting and merging of bundles.

At the event point p_i the algorithm processes the intersections witnessed by p_i, updates the sweepline from ℓ_{i-1} to ℓ_i, and makes the necessary changes (splits or merges) to bundles. The algorithm maintains the invariant that all the red-blue intersections whose witness is to the left of the current event point p_i are already encountered. This results in the following.

Theorem 2 ([12]) *The Mantler-Snoyeink algorithm runs in $O(n \log n)$ time, requires $O(n)$ space, and encounters $O(n)$ bundle-bundle intersections in total.*

2.2 Building the life table

We run the Mantler-Snoyeink algorithm, making the sweepline status structure partially persistent [7]. This guarantees that each blue bundle, that has appeared during the algorithm, can afterwards be retrieved from the version of the sweepline status at the corresponding moment in the past. We assign each blue bundle B_i a *timestamp* t_i – the first moment B_i was encountered. To distinguish between two distinct bundle-bundle intersections discovered at the same moment t_k, we assign the moment $t_k + \epsilon$ to the intersection with a smaller y coordinate. See Figure 2.

The plane sweep algorithm induces a partial order among the red segments: At any moment, the red segments crossed by the sweepline can be ordered from bottom to top. Since the red segments do not intersect, no two of them may swap their relative position. Let $r_1, \ldots, r_n$ be a total order consistent with the partial order along the sweepline at each moment.

We now build the *life table* of red segments and blue bundles (see Figure 2, bottom). The life table is a graph, that has integers from 0 to n on its y axis; and its x axis is the same as the x axis of the original setting. Each red segment r_i is represented by a horizontal line segment whose y coordinate equals i and whose endpoints' x coordinates coincide with the x coordinates of the endpoints of r_i. Each blue bundle that intersected at least one red bundle, is retrieved from the version of the sweepline status, together with its timestamp. In the table, each blue bundle B_j is represented by a vertical line segment (that could possibly be a point), whose x coordinate is the timestamp of B_j. This vertical segment intersects exactly the segments representing all red segments intersected by bundle B_j. We do not store the blue bundles explicitly, but rather maintain a pointer to the

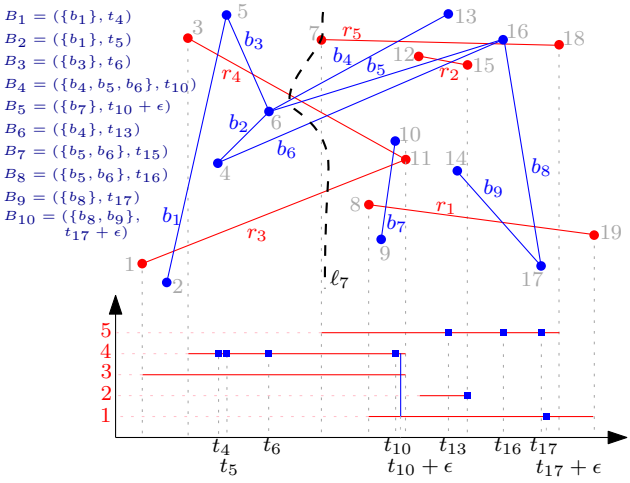


Figure 2: Execution of the algorithm for $R = \{r_1, \ldots, r_5\}$ and $B = \{b_1, \ldots, b_{10}\}$

bundle in (the corresponding version of) the sweepline status structure.

2.3 The resulting data structure

After the life table is built, we sweep it with a horizontal straight line from bottom to top, again making the sweepline status partially persistent. The events now correspond to red segments, and the version of the sweepline status at some moment i contains all blue bundles crossing the horizontal line $y = i$, sorted by x coordinate and stored in a balanced binary tree.

Our ultimate data structure is the persistent sweepline status of the above (second) plane sweep. We are able to retrieve in $O(1)$ time the version of the sweepline status structure at any moment i. The sweepline status at the moment i is a tree storing the blue bundles whose beginning was witnessed before the moment i and whose end was witnessed after that moment (in other words, the blue bundles that intersect the horizontal line $y = i$ in the life table). See Figure 2. Each single bundle is stored in a balanced binary tree, thus the tree of bundles is also a balanced binary tree of height $O(\log n)$, which can be navigated in the same way as a standard binary tree.

Now suppose we wish to search for interesting red-blue intersections on a red segment r_i. Suppose we have an oracle that can tell where the interesting intersection(s) lie with respect to p, for any point p of intersection between r_i and a blue segment. The search is as follows. We retrieve the version of the sweepline status (of the second sweep) at the moment i. This sweepline status is an implicit balanced binary tree, as explained above. We locate the endpoints of r_i in that tree. Then we search in the portion of the tree between r_i's endpoints. The decisions made during the search are made based on our knowledge about the interesting intersections. We conclude.

Theorem 3 *Given a pair R, B of plane straight-line graphs with n edges and vertices in total in both graphs, the RB-Preprocessing problem for R and B can be solved in $O(n \log n)$ time and $O(n)$ space, such that all navigation queries are supported in $O(1)$ time.*

References

[1] M. Abellanas, F. Hurtado, C. Icking, R. Klein, E. Langetepe, L. Ma, B. Palop, and V. Sacristán. The farthest color Voronoi diagram and related problems. In *17th EWCG*, pages 113–116, 2001.

[2] B. Chazelle, H. Edelsbrunner, L. Guibas, and M. Sharir. Algorithms for bichromatic line-segment problems and polyhedral terrains. *Algorithmica*, 11(2):116–132, 1994.

[3] P. Cheillaris, E. Khramtcova, S. Langerman, and E. Papadopoulou. A randomized incremental algorithm for the Hausdorff Voronoi diagram of non-crossing clusters. *Algorithmica*, 2016.

[4] O. Cheong, H. Everett, M. Glisse, J. Gudmundsson, S. Hornus, S. Lazard, M. Lee, and H. Na. Farthest-polygon Voronoi diagrams. *Comput. Geom.*, 44(4):234–247, 2011.

[5] M. Claverol, E. Khramtcova, E. Papadopoulou, M. Saumell, and C. Seara. Stabbing circles for sets of segments in the plane. In *LATIN 2016*, pages 290–305, 2016.

[6] F. Dehne, A. Maheshwari, and R. Taylor. A coarse grained parallel algorithm for Hausdorff Voronoi diagrams. In *ICPP*, pages 497–504. IEEE, 2006.

[7] J. Driscoll, N. Sarnak, D. Sleator, and R. Tarjan. Making data structures persistent. In *Proc. 18th annual ACM symposium on Theory of computing*, pages 109–121. ACM, 1986.

[8] H. Edelsbrunner, L. Guibas, and M. Sharir. The upper envelope of piecewise linear functions: algorithms and applications. *Discr. & Comput. Geom.*, 4(4):311–336, 1989.

[9] L. Guibas, M. Sharir, and S. Sifrony. On the general motion-planning problem with two degrees of freedom. *Discr. & Comput. Geom.*, 4(5):491–521, 1989.

[10] J. Iacono, E. Khramtcova, and S. Langerman. Searching edges in the overlap of two plane graphs. *ArXiv e-prints*, 2017. arXiv:1701.02229.

[11] E. Khramtcova and E. Papadopoulou. Randomized incremental construction for the Hausdorff Voronoi diagram of point clusters. *ArXiv e-prints*, 2016. arXiv:1612.01335.

[12] A. Mantler and J. Snoeyink. Intersecting red and blue line segments in optimal time and precision. In *JCDCG'00, Revised Papers*, pages 244–251. Springer, 2001.

[13] E. Papadopoulou. The Hausdorff Voronoi diagram of point clusters in the plane. *Algorithmica*, 40(2):63–82, 2004.

[14] B. Zhu. Computing the shortest watchtower of a polyhedral terrain in $O(n \log n)$ time. *Comput. Geom.*, 8(4):181–193, 1997.

Straight Skeletons of Monotone Surfaces in Three-Space

Martin Held* Peter Palfrader*

Abstract

We present a simple algorithm to compute the straight skeleton and mitered offset surfaces of a polyhedral terrain in 3D. Like its 2D pedant, the 3D straight skeleton is the result of a wavefront propagation process, which we simulate in order to construct the skeleton in time $\mathcal{O}(n^4 \log n)$, where n is the number of vertices of the terrain. Any mitered offset surface can then be obtained from the skeleton in time linear in the combinatorial size of the skeleton.

1 Introduction

The straight skeleton was introduced to computational geometry over 20 years ago by Aichholzer et al. [2]. Let P be a simple polygon in the plane and consider the following process. At time $t = 0$, each edge of P starts to move towards the interior of P at unit speed in a self-parallel manner, maintaining incidences. The set of moving edges forms a set of polygons $\mathcal{W}_P(t)$, called the wavefront of P at time t. Note that each edge of $\mathcal{W}_P(t)$ is at all times at orthogonal distance t to its corresponding edge of P.

The wavefront needs to be updated at times to remain a set of simple polygons: As edges shrink to zero length (*edge event*), they are removed, and edges are split and incidences updated when a previously non-incident vertex moves into their interior (*split event*). (If the polygon is not in general position then more complex interactions are possible.) The straight skeleton $\mathcal{S}(P)$ of P is then defined as the geometric graph whose edges consist of the traces of wavefront vertices over the propagation process, see Figure 1.

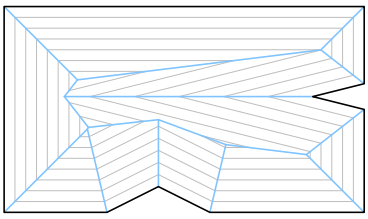


Figure 1: The straight skeleton $\mathcal{S}(P)$ (blue) of an input polygon P (bold) is the union of the traces of the vertices of P as it shrinks. Several instances (wavefronts) of the shrinking polygon are shown in gray.

*Universität Salzburg, FB Computerwissenschaften, 5020 Salzburg, Austria; supported by Austrian Science Fund (FWF) Grant P25816-N15; {held,palfrader}@cosy.sbg.ac.at.

A mitered offset of P at offsetting distance t corresponds to the wavefront at time t. Mitered offsets are inherently linked to the straight skeleton: Given the straight skeleton $\mathcal{S}(P)$ of a polygon P with n vertices, any mitered offset can be constructed in $\mathcal{O}(n)$ time and space [11].

Variations of the straight skeleton problem in the plane have also been investigated, by generalizing the input to arbitrary planar straight line graphs or by adding multiplicative or additive weights to input edges [1, 7, 8, 10].

The algorithm with the currently best worst-case complexity for computing the 2D straight skeleton of an arbitrary simple polygon is due to Eppstein and Erickson [8] and requires both $\mathcal{O}(n^{17/11+\varepsilon})$ time and space, for an arbitrarily small but positive ε. Better runtime bounds can be obtained when restricting the input to monotone polygons. Indeed, Biedl et al. [6] present a simple and easy-to-understand algorithm which requires $\mathcal{O}(n \log n)$ time and linear space to compute the straight skeleton.

Moving to 3-space. Straight skeletons of polytopes were studied by Barequet et al. [5], and recently by Aurenhammer and Walzl [4]. However, while combinatorial complexities have been established for the straight skeleton of polytopes, no runtime bounds have been investigated.

In this work, we consider the straight skeleton and mitered offsets of polyhedral terrains in 3-space. As usual, a polyhedral terrain is a piecewise linear, continuous function of two variables. To simplify matters, we assume that the terrain $\mathcal{T}$ is defined over all of $\mathbb{R}^2$ and that all facets are simply-connected. Furthermore, we assume T is in general position: No more than four supporting planes of the facets of $\mathcal{T}$ shall be tangent to a common sphere and the degree of any vertex of $\mathcal{T}$ shall be at most a constant k.

2 Wavefront Propagation

We consider the wavefront propagation of a polyhedral terrain $\mathcal{T}$. Just as in the plane, where the 2D-wavefront consists of edges at distance t to their corresponding input edge, here the wavefront consists of wavefront facets which are at orthogonal distance t to their corresponding input facets at all times.

Formally, let f be a facet of $\mathcal{T}$, let $\bar{H}_f$ be its supporting plane, and let $\vec{n}_f$ be the unit normal of f with

positive z-coordinate. Then we define the offset supporting plane at distance t to be $H_f(t) \coloneqq \bar{H}_f + t \cdot \vec{n}_f$. The wavefront, just like the input $\mathcal{T}$, is a continuous, piecewise linear surface, i.e., a polyhedral terrain. Its facets at time t are embedded in the offset supporting planes $H_f(t)$ of all facets f of $\mathcal{T}$.

Initially, at time $t = 0$, the wavefront $\mathcal{W}_\mathcal{T}(t)$ is identical to $\mathcal{T}$. When the propagation process starts, all facets of the wavefront move upwards, in positive z-direction. During this propagation, incidences are retained where possible.

For the initial offset at time $t = \delta$, for a sufficiently small $\delta > 0$, retaining the combinatorial structure is possible along edges. Furthermore, locally at vertices of degree three, an offset of the same combinatorial structure is possible. However, at vertices of degree four or more, any offset, even at an infinitesimally small δ, will generally have a combinatorial structure different from the input: The offset surface consists locally of several degree-three vertices that arise from the offsets of the planes incident at the input vertex of higher degree; see Aurenhammer and Walzl [4].

2.1 Events

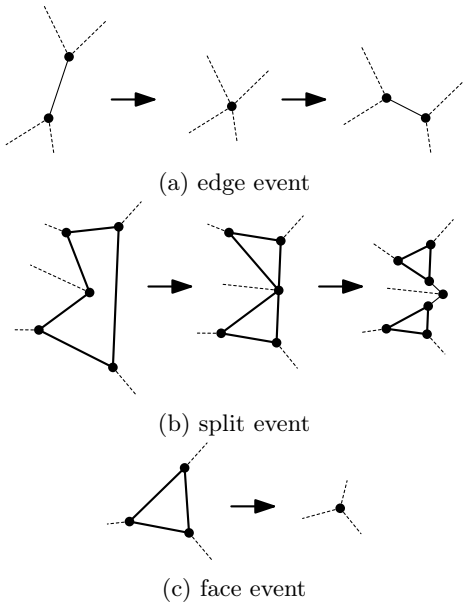
As the wavefront propagation continues, the combinatorial structure of the surface has to be updated and the set of wavefront vertices and their trajectories change at discrete points in time at so-called *events*, when four or more wavefront facets pass through a common point.

Aurenhammer and Walzl [3] consider straight skeletons of polytopes, and they differentiate between events that change the topology of the offset polytope and events that merely change the surface of the polytope. They call the first class *solid events*, which includes *splitting events*, where the polytope disconnects and *piercing events*, where a vertex runs into a facet. However, since our wavefront surface is z-monotone and continuous, these events cannot occur and we will only observe the second class of events, *surface events*, in the wavefront propagation.

An *edge event* happens at time t when an edge of the wavefront collapses to zero length without its incident facets vanishing, too. The two vertices incident at the edge are merged, giving rise to a high-degree vertex. For the wavefront after the event, at time $t+\delta$, this high-degree vertex has to be resolved and generally split again similar to the process at the initial wavefront construction. See Figure 2a.

A second type of event, the *(facet) split event* happens when a vertex v of the wavefront that is incident at facet f moves into the interior of another edge e of f without f collapsing. This case is similar to the split event known from 2D straight skeletons. Combinatorially, the edge e is split at the locus of v and made incident to v, creating a higher-degree vertex which then needs to be resolved again for the post-event wavefront. See Figure 2b.

In the third event type, the *face event*, a facet f may collapse to an empty area. This coincides with one or more edges of f collapsing or a vertex of f moving into the interior of another edge of f. At the event time t, the facet is replaced by a set of edges that cover its boundary without overlapping, thereby merging vertices which now occupy the same locus (if such vertices exist). Again for the post-event wavefront, higher-degree vertices may need to be resolved. See Figure 2c.



(a) edge event

(b) split event

(c) face event

Figure 2: Edge event, split event, and face event during the wavefront propagation.

If the input is not in general position, two vertices that share a facet but not an edge can also meet. This will result in a higher-degree vertex (usually of at least degree six) that needs to be split again. We have ruled out such cases by our general position assumption.

2.2 Computing the Straight Skeleton

Once no more events occur during the propagation process, the process has finished. The three dimensional straight skeleton $\mathcal{S}(\mathcal{T})$ of $\mathcal{T}$ is then the structure whose edges are the traces of wavefront vertices and whose facets are the traces of wavefront edges. To unambiguously refer to features of the 3D straight skeleton, Aurenhammer et al. [4] call the edges of $\mathcal{S}(\mathcal{T})$ *spokes* and its facets *sheets*. The volumes bounded by sheets are called *cells*.

Interior vertices correspond to events that have been observed in the propagation process. Any wavefront vertex or edge remaining in the wavefront at the end of the process induces an unbounded straight skeleton spoke or sheet which continues to infinity.

3 Simulating the Wavefront Propagation

We compute the straight skeleton $\mathcal{S}(\mathcal{T})$ of $\mathcal{T}$ by simulating its wavefront propagation. This requires determining at every stage in the process what the next event will be. To cope with this problem we maintain a priority queue of potential events: As initialization, we first create the initial wavefront for time $t = \delta$, where δ is infinitesimally small, splitting higher degree vertices of $\mathcal{T}$. Then we store for every edge of the wavefront its collapse time, and we store for every vertex of the wavefront the instances of when it will move into any of the edges of its incident facets.

To advance time in our simulation of the wavefront propagation, we fetch the event from the priority queue with the earliest associated time. We process this event by modifying the wavefront combinatorics according to the event type, thereby merging and then splitting vertices as required and as described in the previous section. We add new events to the priority queue for all edges and vertices that were affected or created by the event.

Then, we proceed and fetch the next item from the priority queue. We need to verify that it still is a valid event, that is, we need to check that the edge that is supposed to collapse or the vertex that is supposed to move into an edge are still elements of the wavefront — prior events may have already restructured the wavefront and invalidated this event. If it is a valid event then we process it as described. Otherwise we simply drop it. In either case, this process is repeated until the priority queue is empty.

Number of events. In general, an event happens at point p and time t when four (or more) wavefront facets become incident. (For simplicity reasons, our general position assumption guarantees that no more than four wavefront facets are involved in an event.) This provides a natural upper bound of $\binom{n}{4}$ on the size of the priority queue, where n is the number of facets of the input surface. Based on our experience with different algorithms for computing straight skeletons in the plane, we conjecture that in practice only a small subset of those $\binom{n}{4}$ combinations will be relevant.

Splitting higher-degree vertices. Aurenhammer and Walzl [3] note that an offset surface of a higher-degree vertex v of a three dimensional polytope always exists even though is not necessarily unique. One offset that always exists corresponds to a wavefront where v has been replaced by a tree. In [4], they suggest as a simple approach to enumerate all combinatorially different trees and check whether they correspond to valid offset surfaces of v. The geometry of a tree's element is dictated by its combinatoric properties. Such a valid tree will replace the vertex v in the propagating wavefront.

By our general position assumption, all vertices of the input surface have at most constant degree k. Thus, finding this tree for a single vertex v is a constant-time operation as well. Furthermore, at most a constant number of elements need to be added to the wavefront per input vertex.

Vertex degrees during events. After having constructed the initial wavefront, all moving vertices will be of degree three in the generic case. We investigate the types of vertices that can appear in events.

In an edge event, the edge that connects to degree-three vertices collapses, giving rise to a degree-four vertex v, as shown in Figure 2a. In the generic case, v will have to be split (at constant cost) into two new degree-three vertices connected by a new edge. In our general position assumption we stated that no more than four supporting planes of faces may be tangent to a common sphere. Thus, for our input we will always either split v into two, or v will never again participate in an event.

In a split event, a degree-three vertex v comes to lie on previously non-incident wavefront edge e, which is split in two during the event, giving rise to a degree-five vertex (Figure 2b). In the generic case this vertex will be split into three new vertices, each of degree three. Again, by our general position assumption, this will be the case for our input sets.

For face events we can distinguish two sub-types (Figure 3). In one, a triangle facet will collapse as all its incident edges shrink to zero length. This will give immediate rise to a new degree-three vertex which can then propagate. The other type is where a more complex polygon collapses as some of its edges collapse and maybe some vertices become incident at other edges of the polygon. The facet is replaced by one or more edges, and all resulting vertices will be of degree three and can propagate without any need to be split. Note that multiple face collapses happening at the same time may cause an edge that has the same face on both sides. Such an edge is not removed; instead it propagates like any other edge, similar to how ghost vertices propagate in Biedl et al. [7]. This ensures that all faces remain simply connected during the propagation.

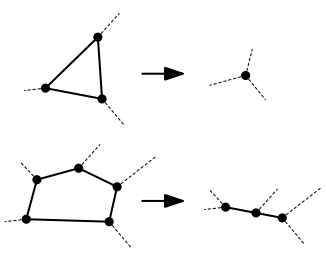


Figure 3: Two types of face collapsing events.

4 Obtaining Offset Surfaces

If only a single mitered offset surface at orthogonal distance t is sought, then one approach to construct this offset is to simply run the wavefront propagation process until time t. Then, the wavefront at this time is the offset surface required.

However, if multiple offset surfaces at different distances should be constructed or if the straight skeleton is already available, then we apply the following process to obtain an offset surface in time linear in the size of the straight skeleton:

For a given spoke s, we denote by $s(t)$ the three dimensional point obtained by intersecting s with a plane at distance t and parallel to the base of any one of its incident cells. Equivalently, $s(t)$ is the location at time t of the wavefront vertex that traced out s.

For every spoke s of the skeleton which exists at (orthogonal) offsetting distance t, i.e., for which $s(t)$ exists, and for every cell c incident at s where (s, c) has not been processed before, we construct an offset facet as follows: Let f_1 be one of the sheets of $s_1 := s$ that is on the boundary of c. We walk along the boundary of f_1, moving in the direction of positive z, until we reach another spoke s_2 of f_1 that exists at distance t. Now let f_2 be the other sheet of c incident at s_2 and repeat the walk in f_2 to find a spoke s_3. Eventually, we will return to our initial spoke s_1. Let s_ℓ be the last one before we returned. (Special handling will be required to process the case of infinite elements.) The polygon with vertices $s_1(t), s_2(t), \ldots, s_\ell(t)$ is now a valid offset facet and we add it to the offset surface we are constructing. We then mark $(s_1, c), (s_2, c), \ldots, (s_\ell, c)$ as processed and continue with our main loop. Once all spokes have been processed, the set of offset facets represents the complete offset surface. This algorithm can be implemented such that all offset facets together with their adjacency relations are obtained.

The correctness of this approach hinges on the property that all offset facets are simple polygons and contain no holes. This property stems from the fact that the wavefront propagation does not experience any piercing event since $\mathcal{T}$ is a terrain.

5 Discussion

We have presented a simple algorithm to compute the straight skeleton of a z-monotone surface. The processing cost of each event is constant for generic input, and the number of events is bounded by $\binom{n}{4}$. We do not expect this bound to be tight, though. Maintaining the events in a priority queue results in a runtime bound of $\mathcal{O}(n^4 \log n)$. Better upper bounds are currently under investigation. A construction by Held [9] establishes an $\Omega(n^2)$ lower bound on the combinatorial complexity of $\mathcal{S}(\mathcal{T})$ for a terrain $\mathcal{T}$. His construction can be adapted to yield the same bound for the combinatorial complexity of one mitered offset.

For descriptive simplicity, our general-position assumption bounds the maximum degree of a vertex that may appear in the propagating wavefront by a small constant. However, using larger constants does not change the process significantly and only results in more complex event handling requirements.

Furthermore, we can relax the bound on the maximum degree of vertices of the input surface. Resolving higher-degree vertices where the degrees are not bound by a constant for the initial wavefront will require more than constant work, but at least for pointed vertices, where all incident faces are confined to one half space, offsetting can be reduced to computing weighted 2D straight skeletons [5] which are well studied [7] and for which implementations exist [10, 11]. Vertices that are saddle-points can still be handled by one of the methods described by Aurenhammer and Walzl [4].

References

[1] O. Aichholzer and F. Aurenhammer. Straight Skeletons for General Polygonal Figures in the Plane. In *Voronoi's Impact on Modern Sciences II*, volume 21, pages 7–21. 1998.

[2] O. Aichholzer, F. Aurenhammer, D. Alberts, and B. Gärtner. A Novel Type of Skeleton for Polygons. *J. Univ. Comp. Sci*, 1(12):752–761, 1995.

[3] F. Aurenhammer and G. Walzl. Three-dimensional straight skeletons from bisector graphs. In *5th Int. Conf. Analytic Number Theory & Spatial Tessellations*, 2013.

[4] F. Aurenhammer and G. Walzl. Straight Skeletons and Mitered Offsets of Nonconvex Polytopes. *DCG*, 56(3):743–801, 2016.

[5] G. Barequet, D. Eppstein, M. T. Goodrich, and A. Vaxman. Straight Skeletons of Three-Dimensional Polyhedra. In *ESA 2008*, pages 148–160.

[6] T. Biedl, M. Held, S. Huber, D. Kaaser, and P. Palfrader. A Simple Algorithm for Computing Positively Weighted Straight Skeletons of Monotone Polygons. *IPL*, 115(2):243–247, 2015.

[7] T. Biedl, M. Held, S. Huber, D. Kaaser, and P. Palfrader. Weighted Straight Skeletons in the Plane. *CGTA*, 48(2):120–133, 2015.

[8] D. Eppstein and J. Erickson. Raising Roofs, Crashing Cycles, and Playing Pool: Applications of a Data Structure for Finding Pairwise Interactions. *DCG*, 22(4):569–592, 1999.

[9] M. Held. On Computing Voronoi Diagrams of Convex Polyhedra by Means of Wavefront Propagation. In *CCCG 1994*, pages 128–133.

[10] M. Held and P. Palfrader. Additive Weights for Straight Skeletons. In *EuroCG 2016*.

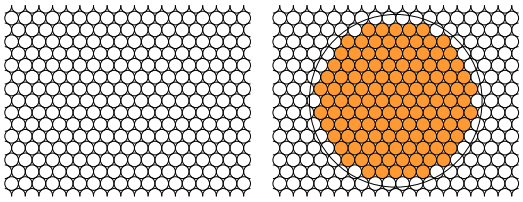
[11] P. Palfrader and M. Held. Computing Mitered Offset Curves Based on Straight Skeletons. *Comp.-Aided Design & Appl.*, 12(4):414–424, Feb. 2015.

Polynomial Time Approximation Schemes for Circle Packing Problems

Helmut Alt* Nadja Scharf†

Abstract

We consider the algorithmic problems of packing a maximum number of unit circles into a given circle of radius r and, vice versa, finding a minimum radius circle into which a given number n of unit circles can be packed. Because of the small size of the input these problems are computationally hard, where the best known algorithms have doubly exponential runtime. We give PTAS's for both problems, which are based on quite simple ideas. The difficulty lies in proving their correctness.



(a) Densest packing of the plane with circles.

(b) Colored circles of the grid lie completely in the container circle.

Figure 1

1 Introduction

We will consider the algorithmic question of densely packing unit circles into a given finite size container. More precisely, our container will be a circle of a given radius r, but the idea of our algorithm can be extended to other shapes of containers (squares, homothets of a fixed convex figure, ...), as well. We will also consider the reverse problem of finding a container of minimum radius for a given number n of unit circles. The problem of packing circles efficiently into a circular container for example arises when installing wires through a tube.

Packing problems have been investigated extensively in operations research (see e.g. [7] for an overview) and mathematical optimization (see e.g. the references in [1]). In particular, the minimum size of a container for packing n unit circles into circles, squares, or other containers is mostly unknown, even for fixed numbers n (see e.g. [1, 2]).

Observe, that our input is just the rational number r, where numerator and denominator are given in binary. For the reverse problem, it will be just the binary representation of the number n of unit circles. Because of this concise input, the computational complexities of the problems seem to be considerably high. We do not know whether they are in NP nor whether they are NP-hard. However, as we shall see, the decision problem whether n unit circles can be packed into a circle of radius r can be formulated as a formula in the existential first order logic of the reals ($\exists\mathbb{R}$). However, the size of this formula is already exponential in the size of the input, so we only know that the problem is in EXPSPACE and, thus, it can be solved in doubly exponential time. Doubly exponential time can also be achieved for the reverse problem, since it can be described by a Tarski formula. But it is not known how to formulate it as a merely existential formula.

The problem of packing the 2- or 3-dimensional space with unit circles or spheres as densely as possible has been considered for centuries. For three dimensions, the problem was first posed in 1611 by Kepler and quite recently solved by Hales et al. [6]. For the significantly easier two-dimensional problem, the densest circle packing is the one where the centers of the circles form a regular triangular lattice as shown in Figure 1a. A first proof is due to Thue in 1890 and a complete and rigorous one was given by Fejes Tóth in 1942 [8]. This packing yields a density of $\rho \approx 0.90690$, meaning that a fraction of ρ of the plane is covered by circles.

In this paper, we consider the problem of approximating the optimal solution for both problems mentioned before. In fact, we will give polynomial time approximation schemes (PTAS) for the problems, based on the simple idea of just intersecting the given container with the optimal packing of the plane (see Figure 1b for an example). Thus, for a given $\varepsilon > 0$ for large (depending on ε) r or n, this gives a $(1-\varepsilon)$- or $(1+\varepsilon)$-approximation respectively. For small values ($r < \frac{c_1}{\varepsilon^2}$ or $n > \frac{c_2}{\varepsilon^2}$ for some constants c_1, c_2), we solve the problems by the exact algorithms and therefore in time exponential in $\frac{1}{\varepsilon}$. Observe, that the simple idea of checking the circles in a suitable neighborhood of the container C one by one whether they lie inside C does not give a polynomial time al-

*Institut für Informatik, Freie Universität Berlin, alt@mi.fu-berlin.de

†Institut für Informatik, Freie Universität Berlin, nadja.scharf@fu-berlin.de

gorithm since their number is already exponential in the size of the input.

2 Preliminaries

Let us first consider the problem of finding the maximum number $n_{\max}$ of unit circles to be packed into a container of a given radius r. $n_{\max}$ can be computed brute force. For the beginning, we will show how the decision problem whether a given number k of unit circles fit into the container circle with radius r can be written as a $\exists\mathbb{R}$-formula. We assume that r is given as fraction of positive integers $\frac{p}{q}$. The variables of the formula are the coordinates of the circle centers. The first part describes that no two circles overlap, i.e. their centers have a distance of at least 2. The second part denotes that the circles lie inside the container circle centered at the origin.

$$\exists\,(x_1,\ldots,x_k,y_1,\ldots,y_k) \bigwedge_{i,j\in\{1,\ldots k\},i\neq j} (x_i-x_j)^2+(y_i-y_j)^2-4\geq 0 \wedge \bigwedge_{i=1}^{k}\left(x_i^2+y_i^2-(r-1)^2\leq 0\right) \quad (1)$$

Basu, Pollack and Roy gave an algorithm to decide the truth of such an $\exists\mathbb{R}$-formula of runtime $s^{v+1}d^{\mathcal{O}(v)}\mathcal{O}(l)\,\mathrm{M}\left(bd^{\mathcal{O}(v)}\right)$, where s is the number of polynomials in the formula, v is the number of variables, d is the maximal degree of the polynomials, b is the bit-size of the coefficients, l is the length of the formula and $\mathrm{M}(m)$ is the runtime for the multiplication of two m-bit integers. In our formula $s=k^2, v=2k, d=2, b=\mathcal{O}(L)$, where L is the bit-size of the representation of r, and $l=\mathcal{O}\left(k^2L\right)$, since there are k^2 polynomials with constant number of coefficients.

In order to solve the maximization problem, we can do a linear (or binary) search for $n_{\max}\in\left\{1,\ldots,\lfloor r^2\rfloor\right\}$ solving the decision problem by Basu et al.'s algorithm in each step. A detailed analysis shows that an overall runtime of this brute-force algorithm of $2^{\mathcal{O}\left(r^2\log r\right)}L^3$ is possible.

3 Algorithm

Algorithm 1 is our approximation algorithm that uses the brute-force algorithm stated above.

To be able to analyze it, we first of all prove an upper bound for $n_{\max}$. Therefore, we need the following lemma.

Lemma 1 *Let S be the set of center points indicating a packing of $n_{\max}$ unit circles inside a circle C with radius r centered at the origin. Let D be the Delaunay triangulation of S. Let t be any triangle of D with at least one vertex of distance at most $r-3$ from the origin. Then all internal angles of t are at most $\frac{2\pi}{3}$.*

Algorithm 1:

Input: Number $r>3$ given as $\frac{p}{q}$, parameter $\varepsilon>0$ given as $\frac{u}{v}$, with p,q,u,v positive integers
Output: Nonnegative integer n_{approx}

1. **if** $r<\frac{40}{3}\cdot\frac{1}{\varepsilon}$ **then**
2. Compute n_{approx} with brute-force algorithm;
3. **end**
4. **else**
5. compute the number $k=2\left(\lfloor\log p\rfloor+1-\lfloor\log q\rfloor\right)+3$ of bits needed for sufficient precision;
6. compute some a with $\pi\geq a\geq\pi-2^{-k}$;
7. compute some b with $\sqrt{12}\leq b\leq\sqrt{12}+2^{-k}$;
8. $n_{\text{approx}}=\left\lceil\frac{a(r-3)^2}{b}\right\rceil$
9. **end**
10. return n_{approx}

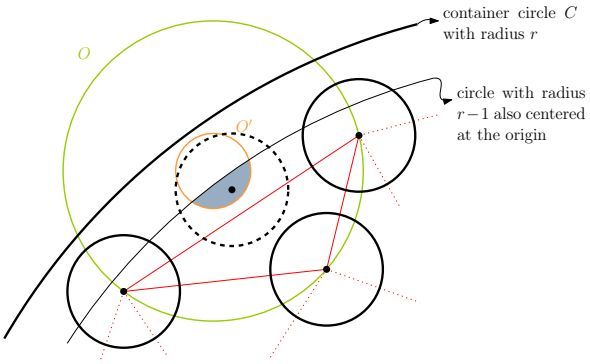


Figure 2

Proof. Let $\triangle\in D$ be a triangle, θ its largest inner angle and O its circumcircle with center o and radius u. Let O' be the circle with radius $u'=\max(u-2,0)$ and center o (see Figure 2). Note that either $u'=0$ or O' has a distance bigger than $r-1$ to the origin since otherwise we could place another unit circle with its center inside O' contradicting the optimality of S. For an example see Figure 2.

Now, we are going to prove that all vertices of triangles with $\theta>\frac{2\pi}{3}$ have distance bigger than $r-3$ from the origin. Since $\theta>\frac{2\pi}{3}$, the smallest internal angle is less than $\frac{\pi}{6}$. All edges of $\triangle$ have length at least 2, so we get by using the sine law

$$u>\frac{2}{\sin\frac{\pi}{6}}=2.$$

This implies that $u'>0$. Therefore, as mentioned above, O' must have distance bigger than $r-1$ from the origin. Hence, O has by construction a distance of bigger than $r-3$ from the origin. Since all vertices of

$\triangle$ lie on O, they also have distance bigger than $r-3$ from the origin. This concludes the proof. □

Now, we can prove the following upper bound for $n_{\max}$. Notice that for $0 \leq r \leq 3$, $n_{\max}$ is known [1].

Lemma 2 *For $r > 3$ it holds that $n_{\max} \leq \frac{\pi}{\sqrt{12}} r^2 + 6r$.*

Proof. Let S be the set of center points indicating a packing of $n_{\max}$ unit circles inside a circle C with radius r centered at the origin. Let D be the Delaunay triangulation of S. In each triangle at most the area of the size of half a unit circle is covered since the sum of inner angles is π. The area of a triangle with largest inner angle $\theta \leq \frac{2\pi}{3}$ and all edge-lengths at least two is at least

$$\frac{1}{2} \cdot 2 \cdot 2 \cdot \sin\theta \geq 2 \cdot \frac{\sqrt{3}}{2} = \sqrt{3}.$$

Hence, at least $\frac{\pi}{2\sqrt{3}}$ of the total area of these triangles is covered by unit circles. The union of all $t \in D$ is the convex hull of S and therefore a circle with radius $r-3$ is covered by triangles, since otherwise there could another circle be packed nonoverlapping inside the container with its center outside of the convex hull. Together with Lemma 1, this gives us that a circle of radius $r-3$ is covered by triangles with largest inner angle $\theta \leq \frac{2\pi}{3}$. Therefore, the total area of all triangles with largest inner angle $\theta \leq \frac{2\pi}{3}$ is at least $\pi (r-3)^2$.

For the rest of the container circle we take 1 as an upper bound for the density of the covered part. In total we get

$$\begin{aligned} n_{\max} &\leq \frac{\pi (r-3)^2 \rho}{\pi} + \frac{\pi \left(r^2 - (r-3)^2\right)}{\pi} \\ &\leq \frac{\pi}{\sqrt{12}} r^2 + 6r \end{aligned}$$

□

Some ideas for the last two proofs are taken from [5].

The following Lemma shows a lower bound of $n_{\max}$. We will use it later to show, that the number of circles returned by our algorithm can be surely packed into the container circle.

Lemma 3 $n_{\max} \geq \left\lceil \frac{\pi (r-3)^2}{\sqrt{12}} \right\rceil$.

Proof. Consider the dense triangular packing of the plane with unit circles. The vertices of every triangle of the grid at least partially contained in the circle with radius $r-3$ have distance at most two from the border of the circle with radius $r-3$. Therefore, they have distance at least 1 from the border of the container circle, i.e. unit circles can be packed with their centers on the vertices of the triangles.

Hence, if we divide the area of the circle with radius $r-3$ by the area of a triangle, we get at a lower bound for twice the number of circles that can be packed on the grid. □

Using Lemma 2 and 3, we can show the following theorem.

Theorem 4 *Algorithm 1 is a PTAS for circle packing.*

Proof. We first show, that Algorithm 1 computes a $(1-\varepsilon)$-approximation for the number of circles that can be packed. Afterwards, we analyze the bit-complexity of the algorithm.

To compute the approximation factor of the algorithm, we first determine the bounds for a and b.

Since $k = 2\left(\lfloor \log p \rfloor + 1 - \lfloor \log q \rfloor\right) + 3$, we get

$$\begin{aligned} a &\geq \pi - 2^{-k} \geq \pi - \frac{1}{8r^2}, \\ b &\leq \sqrt{12} + 2^{-k} \leq \sqrt{12} + \frac{1}{8r^2}. \end{aligned}$$

Now we are going to use these bounds to calculate a lower bound on n_{approx}.

$$n_{\text{approx}} \geq \left\lceil \frac{a (r-3)^2}{b} \right\rceil,$$

since either n_{approx} is the optimal number of circles or this value is returned. Next, we use the bounds for a and b:

$$n_{\text{approx}} \geq \frac{\pi - \frac{1}{8r^2}}{\sqrt{12} + \frac{1}{8r^2}} (r-3)^2$$

which can be shown to satisfy

$$n_{\text{approx}} > \frac{\pi}{\sqrt{12}} (r-3)^2 - 1 \qquad \text{for } r > 1.$$

If n_{approx} is not computed with the brute-force-algorithm, we also now that

$$n_{\text{approx}} \leq \frac{\pi}{\sqrt{12}} (r-3)^2,$$

since $a \leq \pi$ and $b \geq \sqrt{12}$. So following Lemma 3 $n_{\text{approx}} \leq n_{\max}$ and therefore n_{approx} circles can be surely packed into the container circle.

With the lower bound on n_{approx} and using Lemma 2, we can compute the approximation factor of Algorithm 1:

$$\frac{n_{\text{approx}}}{n_{\max}} \geq \frac{\frac{\pi}{\sqrt{12}} (r-3)^2 - 1}{\frac{\pi}{\sqrt{12}} r^2 + 6r} \geq 1 - \varepsilon \quad \text{, since } r \geq \frac{40}{3\varepsilon}.$$

It remains to analyze the running time of Algorithm 1. Let L be the bit-length of the input, i.e. the total length of the encodings of p, q, u, v. The condition for the if in line 1 can be computed by calculating $3pv < 40qu$ which can be done in $\mathcal{O}\left(L^2\right)$ time.

If the result is true, the brute force algorithm is executed which needs $\mathcal{O}\left(2^{\mathcal{O}\left(1/\varepsilon^2 \log(1/\varepsilon)\right)} L^3\right)$ time as described in section 2. Observe, that this bound depends only polynomially on the length L of the input.

If the result is false, first the algorithm computes k. This can be done in $\mathcal{O}(L)$ time since $\lfloor \log p \rfloor + 1$ is the bit length of p and therefore we can compute $\lfloor \log p \rfloor + 1$ and $\lfloor \log q \rfloor$ by counting.

a in line 6 can be computed by calculating π with precision k, i.e. calculating the binary representation of π up to the bit representing $\frac{1}{2^k}$. In the same way, we can compute b by calculating $\sqrt{12}$ up to the bit representing $\frac{1}{2^k}$ and then adding $\frac{1}{2^k}$. These calculations can be done in $\mathcal{O}\left(k^2\right)$ time (see e.g. [4]). Since $k = \mathcal{O}(\log r)$, the running time to compute a and b is in $\mathcal{O}\left(L^2\right)$.

The computation of n_{approx} in line 8 is then a constant number of arithmetic operations on at most L-bit integers, so it is possible in $\mathcal{O}\left(L^2\right)$ time.

In total, the running time is determined by the brute force algorithm. Since this running time is polynomial in L, Algorithm 1 is a PTAS for circle packing. □

4 The reverse problem

Now, we assume that we are given some natural number n and want to determine the minimum radius $r_{\min}$ of a circle into which n unit circles can be packed. Lemmas from the previous section can be used to derive a PTAS for this problem. We only give the basic idea without detailed constants. By Lemmas 2 and 3 we know that a container circle of radius r exists with

$$\rho r^2 - c_1 r \leq n \tag{2}$$

and if a circle of radius r contains n unit circles, then

$$n \leq \rho r^2 + c_2 r \tag{3}$$

for $\rho = \frac{\pi}{\sqrt{12}}$ and $c_1, c_2 > 0$ are constants. Inequality (2) yields that a circle of radius

$$r_{\text{approx}} = d_1 + \sqrt{\frac{n}{\rho} + d_2} \tag{4}$$

for suitable constants $d_1, d_2 \geq 0$ can contain all n circles. Like in the previous section, we compute (an approximation for) this value r_{approx} from n by an algorithm polynomial in the binary size of n. Furthermore, from inequality (3) it follows that for the radius $r_{\min}$ of an optimal container it holds that

$$r_{\min} \geq \sqrt{\frac{n}{\rho} + e_1} - e_2 \tag{5}$$

for constants $e_1, e_2 \geq 0$. By (4) and (5) we have $r_{\text{approx}} - r_{\min} \leq g$ for some constant $g \geq 0$ and therefore,

$$r_{\text{approx}} \leq r_{\min}\left(1 + \frac{g}{r_{\min}}\right).$$

In order to make the relative error $\frac{g}{r_{\min}}$ less than a given ε, it suffices by (5) to have $n \geq \frac{h}{\varepsilon^2}$ for some constant h. For these large n our algorithm returns the value r_{approx}, for smaller n, it solves the problem brute force exactly by solving the corresponding Tarski formula, namely computing the solution of

$$\exists r \left(\phi_1(r) \wedge \forall \left(r' < r\right) \neg \phi_1(r')\right), \tag{6}$$

where ϕ_1 is the formula from (1) (see [3]). Observe, that the size of the latter formula depends only on ε, since $n < \frac{h}{\varepsilon^2}$ and so the runtime for solving (6) does, whereas for large n the algorithm to compute r_{approx} has a runtime polynomial in the bit-size of n. Thus, we have a PTAS for the reverse problem.

References

[1] `www.packomania.com`. Accessed: 2017-01-06.

[2] `http://www2.stetson.edu/~efriedma/packing.html`. Accessed: 2017-01-06.

[3] Saugata Basu, Richard Pollack, and Marie-Françoise Roy. On the combinatorial and algebraic complexity of quantifier elimination. *J. ACM*, 43(6):1002–1045, 1996.

[4] Richard Brent and Paul Zimmermann. *Modern Computer Arithmetic*. Cambridge University Press, 2010.

[5] Hai-Chau Chang and Lih-Chung Wang. A Simple Proof of Thue's Theorem on Circle Packing, 2010, arXiv:1009.4322.

[6] Thomas Hales, Mark Adams, Gertrud Bauer, Dat Tat Dang, John Harrison, Truong Le Hoang, Cezary Kaliszyk, Victor Magron, Sean McLaughlin, Thang Tat Nguyen, Truong Quang Nguyen, Tobias Nipkow, Steven Obua, Joseph Pleso, Jason Rute, Alexey Solovyev, An Hoai Thi Ta, Trung Nam Tran, Diep Thi Trieu, Josef Urban, Ky Khac Vu, and Roland Zumkeller. A formal proof of the Kepler conjecture, 2015, arXiv:1501.02155.

[7] Mhand Hifi and Rym M'hallah. A literature review on circle and sphere packing problems: models and methodologies. *Advances in Operations Research*, 2009:Article ID 150624, 22 p.–Article ID 150624, 22 p., 2009.

[8] L. Fejes Tóth. Über die dichteste Kugellagerung. *Math. Zeit.*, 48:676–684, 1942.

Subquadratic Algorithms for Algebraic Generalizations of 3SUM

Luis Barba[1], Jean Cardinal*[2], John Iacono†[3], Stefan Langerman‡[2], Aurélien Ooms§[2], and Noam Solomon¶[4]

[1]Department of Computer Science, ETH Zürich, Switzerland, luis.barba@inf.ethz.ch
[2]Département d'Informatique, ULB, Belgium, {jcardin,slanger,aureooms}@ulb.ac.be
[3]Department of Computer Science and Engineering, NYU, USA, eurocg2017@johniacono.com
[4]School of Computer Science, TAU, Israel, noam.solom@gmail.com

Abstract

The 3SUM problem asks if an input n-set of real numbers contains a triple whose sum is zero. We consider the 3POL problem, a natural generalization of 3SUM where we replace the sum function by a constant-degree polynomial in three variables. The motivations are threefold. Raz, Sharir, and de Zeeuw gave an $O(n^{11/6})$ upper bound on the number of solutions of trivariate polynomial equations when the solutions are taken from the cartesian product of three n-sets of real numbers. We give algorithms for the corresponding problem of counting such solutions. Grønlund and Pettie recently designed subquadratic algorithms for 3SUM. We generalize their results to 3POL. Finally, we shed light on the General Position Testing (GPT) problem: "Given n points in the plane, do three of them lie on a line?", a key problem in computational geometry.

We prove that there exist bounded-degree algebraic decision trees of depth $O(n^{\frac{12}{7}+\varepsilon})$ that solve 3POL, and that 3POL can be solved in $O(n^2(\log\log n)^{\frac{3}{2}}/(\log n)^{\frac{1}{2}})$ time in the real-RAM model. Among the possible applications of these results, we show how to solve GPT in subquadratic time when the input points lie on $o((\log n)^{\frac{1}{6}}/(\log\log n)^{\frac{1}{2}})$ constant-degree polynomial curves. This constitutes the first step towards closing the major open question of whether GPT can be solved in subquadratic time. To obtain these results, we generalize important tools — such as batch range searching and dominance reporting — to a polynomial setting. We expect these new tools to be useful in other applications. Preprint available on arXiv [5].

*Supported by the "Action de Recherche Concertée" (ARC) COPHYMA, convention number 4.110.H.000023.

†Research partially completed while on sabbatical at the Algorithms Research Group of the Département d'Informatique at ULB with support from a Fulbright Research Fellowship, the Fonds de la Recherche Scientifique — FNRS, and NSF grants CNS-1229185, CCF-1319648, and CCF-1533564.

‡Directeur de recherches du F.R.S.-FNRS.

§Supported by the Fund for Research Training in Industry and Agriculture (FRIA).

¶Supported by Grant 892/13 from the Israel Science Foundation.

1 Introduction

The 3SUM problem is defined as follows: given n distinct real numbers, decide whether any three of them sum to zero. A popular conjecture is that no $O(n^{2-\delta})$-time algorithm for 3SUM exists. This conjecture has been used to show conditional lower bounds for problems in P, notably in computational geometry with problems such as GeomBase, general position [14] and Polygonal Containment [6], and more recently for string problems such as Local Alignment [2] and Jumbled Indexing [4], as well as dynamic versions of graph problems [1, 20], triangle enumeration and Set Disjointness [17]. For this reason, 3SUM is considered one of the key subjects of an emerging theory of complexity-within-P, along with other problems such as all-pairs shortest paths, orthogonal vectors, boolean matrix multiplication, and conjectures such as the Strong Exponential Time Hypothesis [3, 7, 16].

Because fixing two of the numbers a and b in a triple only allows for one solution to the equation $a+b+x=0$, an instance of 3SUM has at most n^2 solution triples. An instance with a matching lower bound is for example the set $\{\frac{1-n}{2},\ldots,\frac{n-1}{2}\}$ (for odd n) with $\frac{3}{4}n^2+\frac{1}{4}$ solution triples. One might be tempted to think that the number of solutions to the problem would lower bound the complexity of algorithms for the decision version of the problem, as it is the case for restricted models of computation [12]. This is a common misconception. Indeed, Grønlund and Pettie [15] recently proved that there exist $\tilde{O}(n^{3/2})$-depth linear decision trees and $o(n^2)$-time real-RAM algorithms for 3SUM.

A natural generalization of the 3SUM problem is to replace the sum function by a constant-degree polynomial in three variables $F\in\mathbb{R}[x,y,z]$ and ask to determine whether there exists any triple (a,b,c) of input numbers such that $F(a,b,c)=0$. We refer to this problem as the *3POL problem*.

For the particular case $F(x,y,z)=f(x,y)-z$ where $f\in\mathbb{R}[x,y]$ is a constant-degree bivariate polynomial, Elekes and Rónyai [10] show that the number of solutions to the 3POL problem is $o(n^2)$ unless f is *special*. Special for f means that f has one of the two special forms $f(u,v)=h(\varphi(u)+\psi(v))$ or

$f(u,v) = h(\varphi(u) \cdot \psi(v))$, where h, φ, ψ are univariate polynomials of constant degree. Elekes and Szabó [11] later generalized this result to a broader range of functions F using a wider definition of specialness. Raz, Sharir and Solymosi [22] and Raz, Sharir and de Zeeuw [21] recently improved both bounds on the number of solutions to $O(n^{11/6})$. They translated the problem into an incidence problem between points and constant-degree algebraic curves. Then, they showed that unless f (or F) is special, these curves have low multiplicities. Finally, they applied a theorem due to Pach and Sharir [19] bounding the number of incidences between the points and the curves. Some of these ideas appear in our approach.

In computational geometry, it is customary to assume the real-RAM model can be extended to allow the computation of roots of constant degree polynomials. We distance ourselves from this practice and take particular care of using the real-RAM model and the bounded-degree algebraic decision tree model with only the four arithmetic operators.

2 Our results

We focus on the computational complexity of 3POL. Since 3POL contains 3SUM, an interesting question is whether a generalization of Grønlund and Pettie's 3SUM algorithm exists for 3POL. If this is true, then we might wonder whether we can beat the $O(n^{11/6}) = O(n^{1.833\ldots})$ combinatorial bound of Raz, Sharir and de Zeeuw [21] with nonuniform algorithms. We give a positive answer to both questions: we show

Theorem 1 *There is a bounded-degree algebraic decision tree of depth $O(n^{\frac{12}{7}+\varepsilon}) = O(n^{1.7143})$ for 3POL. In the real-RAM model, 3POL can be solved in time $O(n^2(\log\log n)^{\frac{3}{2}}/(\log n)^{\frac{1}{2}})$.*

To prove our main result, we present a fast algorithm for the Polynomial Dominance Reporting (PDR) problem, a far reaching generalization of the Dominance Reporting problem. As the algorithm for Dominance Reporting and its analysis by Chan [8] is used in fast algorithms for all-pairs shortest paths, (min,+)-convolutions, and 3SUM, we expect this new algorithm will have more applications.

Our results can be applied to many degeneracy testing problems, such as the General Position Testing (GPT) problem: "Given n points in the plane, do three of them lie on a line?" It is well known that GPT is 3SUM-hard, and it is open whether GPT admits a subquadratic algorithm. Raz, Sharir and de Zeeuw [21] give a combinatorial bound of $O(n^{11/6})$ on the number of collinear triples when the input points are known to be lying on a constant number of polynomial curves, provided these curves are neither lines nor cubic curves. A corollary of our first result is that GPT where the input points are constrained to lie on $o((\log n)^{\frac{1}{6}}/(\log\log n)^{\frac{1}{2}})$ constant-degree polynomial curves (including lines and cubic curves) admits a subquadratic real-RAM algorithm and a strongly subquadratic bounded-degree algebraic decision tree. Interestingly, both reductions from 3SUM to GPT on 3 lines (map a to $(a,0)$, b to $(b,2)$, and c to $(\frac{c}{2},1)$) and from 3SUM to GPT on a cubic curve (map a to (a^3,a), b to (b^3,b), and c to (c^3,c)) construct such special instances of GPT. This constitutes the first step towards closing the major open question of whether GPT can be solved in subquadratic time. Our results also yield efficient algorithms for the problems of counting triples of points spanning unit circles or triangles.

3 Models of Computation

Similarly to Grønlund and Pettie [15], we consider both nonuniform and uniform models of computation. For the nonuniform model, Grønlund and Pettie consider linear decision trees, where one is only allowed to manipulate the input numbers through linear queries to an oracle. Each linear query has constant cost and all other operations are free but cannot inspect the input. In this paper, we consider *bounded-degree algebraic decision trees (ADT)* [23], a natural generalization of linear decision trees, as the nonuniform model. In a bounded-degree algebraic decision tree, one performs constant cost branching operations that amount to test the sign of a constant-degree polynomial for a constant number of input numbers. Again, operations not involving the input are free. For the uniform model we consider the real-RAM model with only the four arithmetic operators.

The problems we consider require our algorithms to manipulate polynomial expressions and, potentially, their real roots. For that purpose, we will rely on Collins cylindrical algebraic decomposition (CAD) [9].

Collins CAD solves any *geometric* decision problem that does not involve quantification over the integers in time doubly exponential in the problem size. This does not harm our results as we exclusively use this algorithm to solve constant size subproblems. Geometric is to be understood in the sense of Descartes and Fermat, that is, the geometry of objects that can be expressed with polynomial equations. In particular, it allows us to make the following computations in the real-RAM and bounded-degree ADT models:

1. Given a constant-degree univariate polynomial, count its real roots in $O(1)$ operations,
2. Given a constant number of univariate polynomials of constant degree, sort their real roots in $O(1)$ operations,
3. Given a point in the plane and an arrangement of a constant number of constant-degree polynomial planar curves, locate the point in the arrangement in $O(1)$ operations.

4 Nonuniform algorithm for explicit 3POL

As a glimpse of our results, we detail here our nonuniform algorithm for explicit 3POL. The other results can be found in the arXiv preprint [5].

Problem (explicit 3POL) *Let $f \in \mathbb{R}[x, y]$ be a bivariate polynomial of constant degree, given three sets A, B, and C, each containing n real numbers, decide whether there exist $a \in A$, $b \in B$, and $c \in C$ such that $c = f(a, b)$.*

Theorem 2 *There is a bounded-degree ADT of depth $O(n^{\frac{12}{7}+\varepsilon})$ for explicit 3POL.*

Idea The idea is to partition the sets A and B into small groups of consecutive elements. That way, we can divide the $A \times B$ grid into cells with the guarantee that each curve $c = f(x, y)$ in this grid intersects a small number of cells. For each such curve and each cell it intersects, we search c among the values $f(a, b)$ for all (a, b) in a given intersected cell. We generalize Fredman's trick [13] — and how it is used in Grønlund and Pettie's paper [15] — to quickly obtain a sorted order on those values, which provides us a logarithmic search time for each cell. Note that it is easy to modify the algorithm to count or report the solutions. In the latter case, the algorithm becomes output sensitive.

$A \times B$ grid partitioning Let $A = \{ a_1 < a_2 < \cdots < a_n \}$ and $B = \{ b_1 < b_2 < \cdots < b_n \}$. For some positive integer g to be determined later, partition the interval $[a_1, a_n]$ into n/g blocks $A_1^*, A_2^*, \ldots, A_{n/g}^*$ such that each block contains g numbers in A. Do the same for the interval $[b_1, b_n]$ with the numbers in B and name the blocks of this partition $B_1^*, B_2^*, \ldots, B_{n/g}^*$. For the sake of simplicity, and without loss of generality, we assume here that g divides n. To each of the $(n/g)^2$ pairs of blocks A_i^* and B_j^* corresponds a cell $A_i^* \times B_j^*$. By definition, each cell contains g^2 pairs in $A \times B$. For the sake of notation, we define $A_i = A \cap A_i^* = \{ a_{i,1} < a_{i,2} < \cdots < a_{i,g} \}$ and $B_j = B \cap B_j^* = \{ b_{j,1} < b_{j,2} < \cdots < b_{j,g} \}$. Figure 1 depicts this construction.

Two useful lemmas follow from this construction:

Lemma 3 *For a fixed value $c \in C$, the curve $c = f(x, y)$ intersects $O(\frac{n}{g})$ cells. Moreover, those cells can be found in $O(\frac{n}{g})$ time.*

Proof. The constant-degree polynomial curve $c = f(x, y)$ is composed of $O(1)$ xy-monotone pieces. Walk and sweep the $A \times B$ grid to locate those pieces. □

Lemma 4 *If the sets A, B, C can be preprocessed in $S_g(n)$ time so that, for any given cell $A_i^* \times B_j^*$ and any given $c \in C$, testing whether $c \in f(A_i \times B_j) = \{ f(a, b) \colon (a, b) \in A_i \times B_j \}$ can be done in $O(\log g)$ time, then explicit 3POL can be solved in $S_g(n) + O(\frac{n^2}{g} \log g)$ time.*

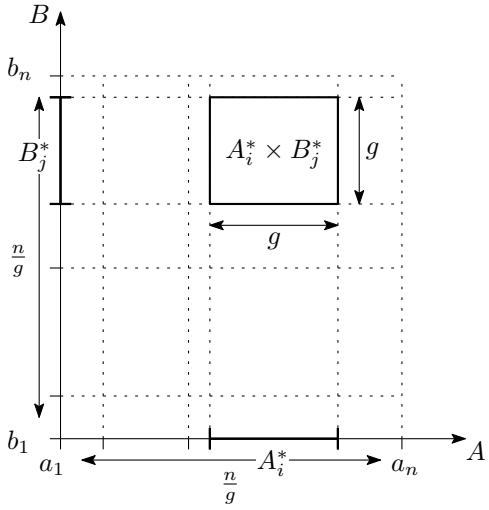


Figure 1: Partitioning A and B.

Remark We do not give a $S_g(n)$-time real-RAM algorithm for preprocessing the input, but only a $S_g(n)$-depth bounded-degree ADT. In fact, this preprocessing step is the only nonuniform part of the algorithm.

Preprocessing All that is left to prove is that $S_g(n)$ is subquadratic for some choice of g. To achieve this we sort the points inside each cell using Fredman's trick [13]. Grønlund and Pettie [15] use this trick to sort the sets $A_i + B_j = \{ a + b \colon (a, b) \in A_i \times B_j \}$ with few comparisons: sort the set $D = (\cup_i [A_i - A_i]) \cup (\cup_j [B_j - B_j])$, where $A_i - A_i = \{ a - a' \colon (a, a') \in A_i \times A_i \}$ and $B_j - B_j = \{ b - b' \colon (b, b') \in B_j \times B_j \}$, using $O(n \log n + |D|)$ comparisons, then testing whether $a + b \leq a' + b'$ can be done using the free (already computed) comparison $a - a' \leq b' - b$. We use a generalization of this trick to sort the sets $f(A_i \times B_j)$. For each B_j, for each pair $(b, b') \in B_j \times B_j$, define the curve $\gamma_{b,b'} = \{ (x, y) \colon f(x, b) = f(y, b') \}$. Define the sets $\gamma^0_{b,b'} = \gamma_{b,b'}, \gamma^-_{b,b'} = \{ (x, y) \colon f(x, b) < f(y, b') \}, \gamma^+_{b,b'} = \{ (x, y) \colon f(x, b) > f(y, b') \}$. The following lemma follows by definition:

Lemma 5 *Given a cell $A_i^* \times B_j^*$ and two pairs $(a, b), (a', b') \in A_i \times B_j$, deciding whether $f(a, b) < f(a', b')$ (respectively $f(a, b) = f(a', b')$ and $f(a, b) > f(a', b')$) amounts to deciding whether the point (a, a') is contained in $\gamma^-_{b,b'}$ (respectively $\gamma^0_{b,b'}$ and $\gamma^+_{b,b'}$).*

There are $N := \frac{n}{g} \cdot g^2 = ng$ pairs $(a, a') \in \cup_i [A_i \times A_i]$ and there are N pairs $(b, b') \in \cup_j [B_j \times B_j]$. Sorting the $f(A_i \times B_j)$ for all (A_i, B_j) amounts to locating each point (a, a') with respect to each curve $\gamma_{b,b'}$. We solve this subproblem in $O(N^{\frac{4}{3}+\varepsilon})$ operations using a

duality lemma and a modified version of the algorithm of Matoušek [18] for Hopcroft's problem.

Analysis Combining this new algorithm with Lemma 4 yields a $O((ng)^{4/3+\varepsilon} + n^2 g^{-1} \log g)$-depth bounded-degree ADT for 3POL. By optimizing over g, we get $g = \Theta(n^{2/7-\varepsilon})$, and the previous expression simplifies to $O(n^{12/7+\varepsilon})$, proving Theorem 2.

References

[1] Amir Abboud and Virginia Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *FOCS*, pages 434–443. IEEE Computer Society, 2014.

[2] Amir Abboud, Virginia Vassilevska Williams, and Oren Weimann. Consequences of faster alignment of sequences. In *ICALP (1)*, volume 8572 of *LNCS*, pages 39–51, 2014.

[3] Amir Abboud, Virginia Vassilevska Williams, and Huacheng Yu. Matching triangles and basing hardness on an extremely popular conjecture. In *STOC*, pages 41–50. ACM, 2015.

[4] Amihood Amir, Timothy M. Chan, Moshe Lewenstein, and Noa Lewenstein. On hardness of jumbled indexing. In *ICALP (1)*, volume 8572 of *LNCS*, pages 114–125, 2014.

[5] Luis Barba, Jean Cardinal, John Iacono, Stefan Langerman, Aurélien Ooms, and Noam Solomon. Subquadratic algorithms for algebraic generalizations of 3SUM. *ArXiv e-prints*, 2016. arXiv:1612.02384 [cs.DS].

[6] Gill Barequet and Sariel Har-Peled. Polygon containment and translational min Hausdorff distance between segment sets are 3SUM-hard. *Int. J. Comput. Geometry Appl.*, 11(4):465–474, 2001.

[7] Marco L. Carmosino, Jiawei Gao, Russell Impagliazzo, Ivan Mihajlin, Ramamohan Paturi, and Stefan Schneider. Nondeterministic extensions of the strong exponential time hypothesis and consequences for non-reducibility. In *ITCS*, pages 261–270. ACM, 2016.

[8] Timothy M. Chan. All-pairs shortest paths with real weights in $O(n^3/\log n)$ time. *Algorithmica*, 50(2):236–243, 2008.

[9] George E. Collins. Hauptvortrag: Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In *Automata Theory and Formal Languages*, volume 33 of *LNCS*, pages 134–183. Springer, 1975.

[10] György Elekes and Lajos Rónyai. A combinatorial problem on polynomials and rational functions. *J. Comb. Theory, Ser. A*, 89(1):1–20, 2000.

[11] György Elekes and Endre Szabó. How to find groups? (and how to use them in Erdős geometry?). *Combinatorica*, 32(5):537–571, 2012.

[12] Jeff Erickson. Lower bounds for linear satisfiability problems. *Chicago J. Theor. Comput. Sci.*, 1999.

[13] Michael L. Fredman. How good is the information theory bound in sorting? *Theor. Comput. Sci.*, 1(4):355–361, 1976.

[14] Anka Gajentaan and Mark H. Overmars. On a class of $O(n^2)$ problems in computational geometry. *Comput. Geom.*, 5:165–185, 1995.

[15] Allan Grønlund and Seth Pettie. Threesomes, degenerates, and love triangles. In *Foundations of Computer Science (FOCS 2014)*, pages 621–630. IEEE, 2014.

[16] Monika Henzinger, Sebastian Krinninger, Danupon Nanongkai, and Thatchaphol Saranurak. Unifying and strengthening hardness for dynamic problems via the online matrix-vector multiplication conjecture. In *STOC*, pages 21–30. ACM, 2015.

[17] Tsvi Kopelowitz, Seth Pettie, and Ely Porat. Higher lower bounds from the 3SUM conjecture. In *SODA*, pages 1272–1287. SIAM, 2016.

[18] Jirí Matoušek. Range searching with efficient hierarchical cutting. *Discrete & Computational Geometry*, 10:157–182, 1993.

[19] János Pach and Micha Sharir. On the number of incidences between points and curves. *Combinatorics, Probability & Computing*, 7(1):121–127, 1998.

[20] Mihai Pătraşcu. Towards polynomial lower bounds for dynamic problems. In *STOC*, pages 603–610. ACM, 2010.

[21] Orit E. Raz, Micha Sharir, and Frank de Zeeuw. Polynomials vanishing on cartesian products: The Elekes-Szabó theorem revisited. In *SoCG*, volume 34 of *LIPIcs*, pages 522–536, 2015.

[22] Orit E. Raz, Micha Sharir, and József Solymosi. Polynomials vanishing on grids: The Elekes-Rónyai problem revisited. In *SoCG*, page 251. ACM, 2014.

[23] Andrew Yao. A lower bound to finding convex hulls. *J. ACM*, 28(4):780–787, 1981.

Computing Triangulations with Minimum Stabbing Number

Victor Alvarez * Sándor P. Fekete * Arne Schmidt *

Abstract

For a given point set P or a polygon $\mathcal{P}$, we consider the problem of finding a triangulation T with minimum stabbing number, i.e., a triangulation such that the maximal number of segments hit by any ray going through T is minimized. We prove that this problem is NP-hard; this differs from the problem of triangulating a polygon with minimum edge weight, which is solvable in polynomial time with a simple dynamic program [7]. In an experimental part we test various heuristics.

1 Introduction

Triangulations of point sets or polygons are natural auxiliary structures for a wide range of applications. Depending on the context, a variety of objective functions have been considered to measure their quality. Arising from the context of ray shooting, one such measure that has received a growing amount of attention is the *stabbing number*: This is the maximum number of triangulation edges any line (called a *stab*) can intersect; finding a triangulation of minimum stabbing number corresponds to finding a triangulation that is as "transparent" or "shallow" as possible. This type of question has been considered for a number of structures on a given point set, such as matchings, trees, or triangulations; see Fekete et al. [6].

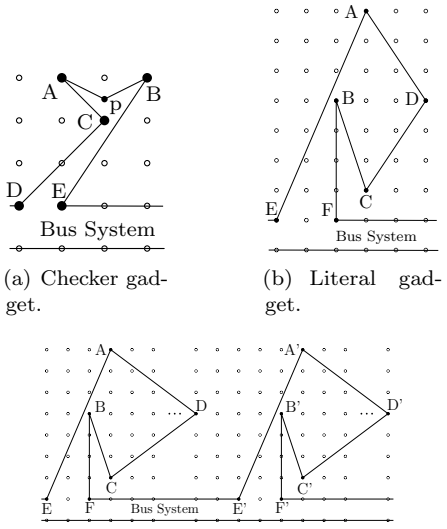
In this paper we consider two variations of stabbing problems: (1) triangulating a point set P or (2) a polygon $\mathcal{P}$ such that the stabbing number is minimized. More formally, the Minimum Stabbing Triangulation Problem (MSTR) asks for a triangulation T of a given point set P, such that the stabbing number $\max_{S\in\mathcal{S}}(|\{e \in T : e \cap S \neq \emptyset\}|)$ is minimal, where $\mathcal{S}$ is the set of all stabs. The problem of triangulating a polygon (MSPT) is defined analogously.

Related Work. Chazelle et al. [3] consider geodesic triangles, i.e., triangles with concave sides. They show that for every polygon P with n vertices, $O(\log n)$ triangles can be hit. Fekete et al. [6] prove NP-hardness of stabbing problems for matchings and trees, and triangulations in [9]. Aichholzer et al. [2] prove NP-hardness of the stabbing problem for polygons. De Berg and van Kreveld [4] study decompositions of rectilinear polygons into rectangles and show that there exists such a decomposition with stabbing number $\mathcal{O}(\log n)$, where the stabbing number counts the number of rectangles any line can intersect. More recently, Piva and de Souza [8] provide a new IP formulation and solve instances with 5000 points [8]. Welzl [11] shows how to construct a spanning tree having a crossing number of $\mathcal{O}(\sqrt{n})$, which is closely related to the stabbing number.

Our Contribution. We prove that the MSPT is NP-hard for axis-parallel stabs; we also present a number of heuristics and experimental results for the MSTR.

2 Triangulating Polygons

Theorem 1 *The problem MSPT with axis-parallel stabs is* NP*-hard.*



(a) Checker gadget.

(b) Literal gadget.

(c) Variable gadget for variable x with x left and $\overline{x}$ right

Figure 1: Components of the polygon with bus system underneath. Circles represent grid points, filled circles are vertices of the polygon.

Proof. We give a reduction of 3SAT to our problem. Consider a 3SAT instance I consisting of C_I clauses, L_I literals and V_I variables. We transform this instance into a polygon that consists of four components (see Figure 1 for an overview): a *literal gadget* for representing a literal l; a *variable gadget* for representing a variable v; a *checker gadget* for guaranteeing a valid assignment of variables and literals; a *bus system* for connecting all gadgets.

*Department of Computer Science, TU Braunschweig, Germany. {s.fekete,arne.schmidt}@tu-bs.de, alvarez@ibr.cs.tu-bs.de

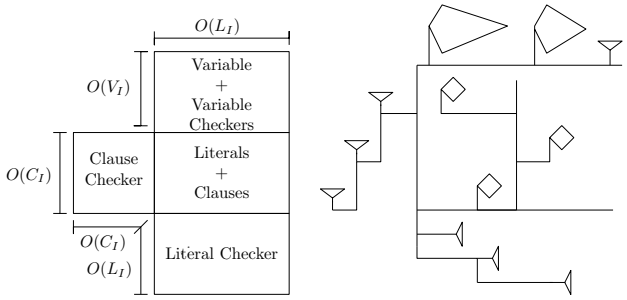


Figure 2: Left: Block model of the 3SAT polygon. Right: Part of a reduction polygon.

The Gadgets. The gadgets are shown in Figure 1. By choosing the diagonals AC or BD we can choose the corresponding literal to be `false` or `true`, respectively. Considering a variable gadget the setting will be the other way round: Choosing AC and $B'D'$ sets the variable to `true` and choosing $A'C'$ and BD sets the variable to `false`. The checker gadgets force a correct setting of all diagonals, e.g., a variable x cannot be `true` and `false` at the same time. By scaling the grid we are allowed to insert more points in the checker gadget and thus, increase the maximum stabbing number.

The Construction. We omit full details due to space constraints. Consider the block model in Figure 2. We place the gadgets corresponding to the blocks; the clauses are represented by literals in the middle, variables are placed on top, literal checkers below and clause checkers to the left. Figure 2 shows part of an example.

Stabbing Number. For all stabs the stabbing number is at most $4C_I$ (proof omitted due to space limitations). Thus, if the 3SAT instance is satisfiable then the stabbing number of the triangulation of the polygon is at most $4C_I$.

We can also show the other direction: if there is a triangulation with stabbing number $\leq 4C_I$, then the 3SAT instance is satisfiable. Finally, it is straightforward to check that the overall construction has polynomial size. □

3 Integer Programm Formulation

When trying to solve the problem with an IP, it is natural to focus on the maximal number of non-crossing edges, but using triangles is the better choice [10]. This IP may have $\Omega(n^3)$ variables, however, we only need to consider empty triangles that do not enclose any point. The basic idea of the IP is that each edge e in a triangulation is part of two triangles if e is not part of the convex hull. If e is a part of the convex hull then there is exactly one triangle having e as an edge.

Definition 1 *We define the edges of the convex hull* $E_H := \{e \in E | e \text{ is on the convex hull}\}$. $\Delta(P)$ *is the set of all empty triangles induced by point set* P. $\delta^-(ij) := \{ijl \in \Delta(P) | ijl \text{ is a right turn}\}$ *and* $\delta^+(ij) := \{ijl \in \Delta(P) | ijl \text{ is a left turn}\}$. *Moreover, for a triangle* ijl *and a stab* S, *let* $c^S_{ijl} := \beta^S_{ij} + \beta^S_{jl} + \beta^S_{il}$,

$$\text{where } \beta^S_{ij} := \begin{cases} 1, & \text{if } ij \in E_H \text{ and } ij \text{ intersects } S \\ 0.5, & \text{if } ij \notin E_H \text{ and } ij \text{ intersects } S \\ 0, & \text{else} \end{cases}$$

With this, we can formulate a triangle based IP for MSTR (analogously for MSPT) as follows:

min K

s.t.:

$$\forall ij \in E \setminus E_H: \sum_{\substack{ijl \in \Delta(P) \\ ijl \in \delta^+(ij)}} x_{ijl} = \sum_{\substack{ijl \in \Delta(P) \\ ijl \in \delta^-(ij)}} x_{ij}$$

$$\forall ij \in E_H: \sum_{ijl \in \Delta(P)} x_{ijl} = 1$$

$$\forall S \in \mathcal{S}: \sum_{\substack{ijl \in \Delta(P): \\ ijl \cap S \neq \emptyset}} c^S_{ijl} x_{ijl} \leq K$$

$$\forall t \in \Delta(P): x_t \in \mathbb{B}$$

4 Heuristics

In addition to exact methods, we also tested a number of heuristics: the new *BER*-algorithm (Section 4.1), local optimization based on *flipping* (Section 4.2), and an algorithm for obtaining a smaller edge set (Section 4.3).

4.1 Bridge Error Rate (BER)

The idea of the *Bridge Error Rate* (BER) algorithm is as follows. For each stab S there is a lower bound on the number of edges crossing S (Theorem 3). If we insert an edge e then we may raise this lower bound. Because we do not want to increase the lower bound, we greedily add edges that raise it as little as possible.

Definition 2 *(Bridge) Consider a stab* S, *as shown in Figure 3.* S *splits the point set* P *into subsets* P_M, P_U *and* P_L, *i.e., points on* S *and in the upper and lower half-space, respectively. For a connected subset* $\texttt{CH}^*(P_U) \subseteq \texttt{CH}(P_U)$, *where* $\texttt{CH}(V)$ *is the convex hull of a set* V, *a point* p *is in* $\texttt{CH}^*(P_U)$ *iff there is a point* $q \in P \setminus P_U$ *such that* $\{p, q\}$ *does not cross* $\texttt{CH}(P_U)$. *Analogously, we define* $\texttt{CH}^*(P_L)$. *The* bridge $B(S)$ *is the union of* $\texttt{CH}^*(P_U)$, $\texttt{CH}^*(P_L)$ *and* P_M. *Furthermore, we define* $|B(S)|$ *as the number of points in* $B(S)$. *Note that* $\texttt{CH}^*$ *may not be an upper or lower envelope of the convex hull.*

Theorem 3 *Consider a stab* S *and its bridge* $B(S)$. *Let* m *be the number of points on* S, *and* m^* *the*

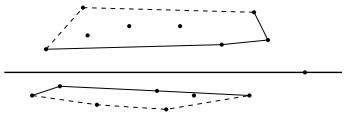


Figure 3: A stab (solid fat line). The solid thin line is $\texttt{CH}^*(P_L)$ and $\texttt{CH}^*(P_R)$, respectively. The dashed lines are the extensions to the convex hull on each side.

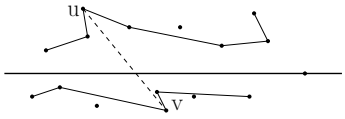


Figure 4: A constrained bridge. The end points of $\{u, v\}$ are connected to $B(S)$ via geodesic paths.

number of points on S that are not part of $\texttt{CH}(P)$. Then there are at least $|B(S)| + m + m^* - 1$ edges intersecting S.

If an arbitrary edge e is added that intersects a stab S and some edges in $B(S)$, we remove these edges from $B(S)$ and add geodesic paths connecting the end points of e and $B(S)$ (see Figure 4). This results in a new *constrained* bridge, yielding a new lower bound. Note that Theorem 3 also holds for the constrained bridge. Having constrained bridges, we can insert one edge after another until we have a triangulation. Further details are omitted due to limited space.

4.2 Flipping

Flipping an edge in a triangulation is a natural heuristic approach to local optimization. We flip edges only when the flip does not increase the stabbing number of any stab, and there is a stab with maximum stabbing number whose number decreases after the flip. If no such edge exists, we flip edges as long as the stabbing number of a stab is not increased and the total edge length decreases. As described in Section 5, with these criteria we achieve near-optimal solutions.

4.3 Smaller Edge Sets

When solving the IP for our problems, the runtime to solve an instance depends on the number of variables, i.e., the number of edges. An idea for using a reduced edge set comes from the IP for triangulations. We begin with the convex hull and add edges from the smallest triangle in terms of edge length. From these new edges we repeat the procedure in the left and right half-space from each supporting line. A reduced edge set can be seen in Figure 5.

5 Experiments

Our experiments were run on 64-bit Ubuntu 14.04.4 LTS with an Intel Core i7-4770 @3.4GHz with 32KB L1 cache, 256KB L2 cache, 8192KB L3 cache and 32GB RAM. We used the g++ compiler version 4.8.4 and optimization flag -O3.

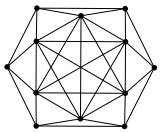


Figure 5: Example for an edge set with 12 points. From 66 possible edges we now only use 33.

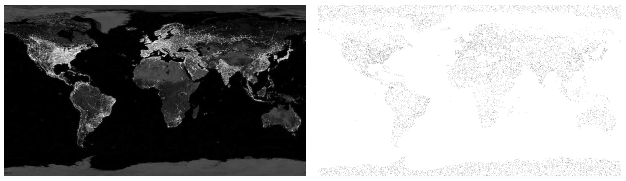


Figure 6: Left: Satellite image of earth by night [1]. Right: Generated image with 10000 points.

solved	# points
30	170
15	190
<15	>190

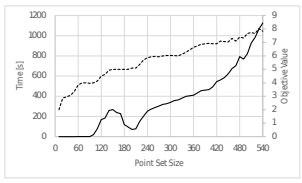


Figure 7: Left: Number of solved instances (out of 30) for different point sizes. Right: Average time and objective value of solvable instances. The dashed line represents the objective value, the solid line the average time in seconds.

Clustered Instances. Clustered instances were generated from a lightmap, similar to the description in Fekete et al. [5]. For a given illumination map, the brightness value induce a density function that can be used for random sampling of points from the image. For the image shown in Figure 6, we created 30 instances per point set size for obtaining *clustered* instances. We omit other instance types.

Optimal Solutions. The left of Figure 7 shows how many instances (out of 30) with n points are solvable within the 30-minute time limit. The graph on the right side shows average runtime and average objective value for each number of points.

Triangulation Heuristics. Figure 8 shows a comparison of the optimum and the BER-algorithm with flipping. The solutions are quite close to the optimum; the maximum difference is 3. We also tested other methods (e.g., Delaunay triangulations) which resulted in slightly worse solutions.

Reduced Edge Sets. Consider Figure 9. We observe that the reduced edge set appears to grow (close to)

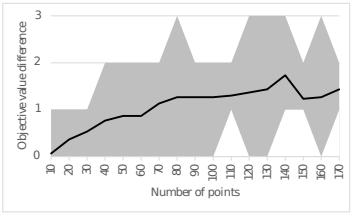


Figure 8: Difference between the optimal objective value and the result of the BER algorithm with flipping. Shaded area shows the range. The solid line shows average difference.

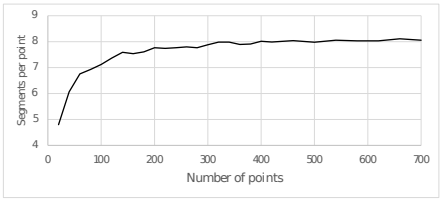


Figure 9: Number of edges per point in the reduced edge set on clustered instances.

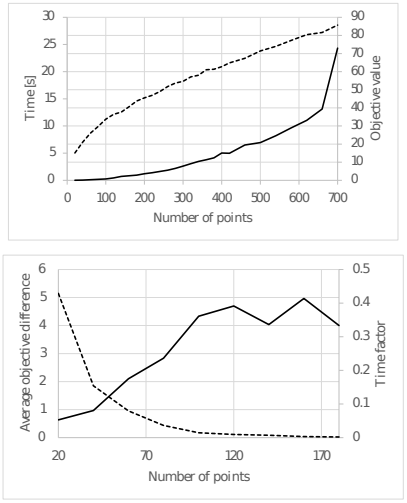


Figure 10: Top: Average time (solid line) and stabbing number (dashed line) using a reduced edge set. Bottom: Difference of stabbing number of an optimal solution and with reduced edge set (solid line) and the time ratio reduced edge set / optimum.

linearly in the number of points. Figure 10 shows a comparison between optimum and the solutions with reduced edge set. The speedup is outstanding; for example, the computing time is reduced by a factor of 500 for 180 points. Note that this comes at the expense getting a suboptimal solution.

6 Future Work

We have shown that the problem MSPT of finding a triangulation of a polygon with minimum stabbing number is NP-hard. Our experiments for the MSTR show that flipping edges can yield excellent solutions. This is also the case for our BER-algorithm; however, its runtime is rather high. We can also use a reduced edge set, resulting in a massive speedup with an only slightly increased stabbing number.

An interesting challenge on the theoretical side lies in developing approximation algorithms, if there are any; this is already an open problem for the other problems of stabbing type considered in [6]. On the experimental side, it is also of interest to collect hard instances, i.e., instances that are practically difficult to solve to optimality.

References

[1] Blick.ch: Lichter der Erde. `http://www.blick.ch/life/wissen/nasa-bilder-aus-dem-all-lichter-der-erde-id2131506.html`. Accessed: 2016-06-21.

[2] O. Aichholzer, F. Aurenhammer, T. Hackl, F. Hurtado, A. Pilz, P. Ramos, J. Urrutia, P. Valtr, and B. Vogtenhuber. On k-convex point sets. *Computational Geometry*, 47(8):809–832, 2014.

[3] B. Chazelle, H. Edelsbrunner, M. Grigni, L. Guibas, J. Hershberger, M. Sharir, and J. Snoeyink. Ray shooting in polygons using geodesic triangulations. *Algorithmica*, 12(1):54–68, 1994.

[4] J. A. de Loera, S. Hosten, F. Santos, and B. Sturmfels. The polytope of all triangulations of a point configuration. *Documenta Mathematica*, 1(4):103–119, 1996.

[5] S. P. Fekete, A. Haas, M. Hemmer, M. Hoffmann, I. Kostitsyna, D. Krupke, F. Maurer, J. S. B. Mitchell, A. Schmidt, C. Schmidt, and J. Troegel. Computing nonsimple polygons of minimum perimeter. In *15th International Symposium on Experimental Algorithms (SEA 2016)*, pages 134–149, 2016.

[6] S. P. Fekete, M. E. Lübbecke, and H. Meijer. Minimizing the stabbing number of matchings, trees, and triangulations. *Discrete & Computational Geometry*, 40(4):595–621, 2008.

[7] G. Klincsek. Minimal triangulations of polygonal domains. *Ann. Discrete Math*, 9:121–123, 1980.

[8] B. Piva and C. C. de Souza. Minimum stabbing rectangular partitions of rectilinear polygons. *Computers & Operations Research*, 80:184–197, 2017.

[9] B. Piva, S. P. Fekete, and C. de Souza. On triangulations with minimum stabbing or minimum crossing number. Manuscript, 2016.

[10] A. Tajima. Optimality and integer programming formulations of triangulations in general dimension. In *International Symposium on Algorithms and Computation*, pages 378–387. Springer, 1998.

[11] E. Welzl. On spanning trees with low crossing numbers. In *Data structures and efficient algorithms*, pages 233–249. Springer, 1992.

Bottleneck Bichromatic Full Steiner Trees

A. Karim Abu-Affash* Sujoy Bhore† Paz Carmi‡ Dibyayan Chakraborty§

Abstract

Given two sets of points in the plane, Q of n (terminal) points and S of m (Steiner) points, where each of Q and S contains bichromatic points (red and blue points), a full bichromatic Steiner tree is a Steiner tree in which all points of Q are leaves and each edge of the tree is bichromatic (i.e., connects a red and a blue point). In the bottleneck bichromatic full Steiner tree (BBFST) problem, the goal is to compute a bichromatic full Steiner tree T, such that the length of the longest edge in T is minimized. In k-BBFST problem, the goal is to find a bichromatic full Steiner tree T with at most $k \leq m$ Steiner points from S, such that the length of the longest edge in T is minimized. In this paper, we present an $O((n+m)\log^2 m)$ time algorithm that solves the BBFST problem. Moreover, we show that k-BBFST problem is NP-hard and we give a polynomial-time 9-approximation algorithm for the problem.

1 Introduction

Given a weighted graph $G = (V, E)$ with $V = Q \cup S$, where Q and S are sets of terminal and Steiner points, respectively, a Steiner tree is an acyclic connected subgraph of G spanning all vertices of Q. Informally, Steiner points are new auxiliary nodes that can be added to the network to improve its performance. In the classical *Steiner tree* problem, the goal is to find a Steiner tree T, such that the length of the edges of T is minimized. This problem has been shown to be NP-complete [6, 14], and for arbitrary weighted graphs, many approximation algorithms have been proposed [8, 16, 17].

In the geometric context, i.e., Q and S are sets of points in the plane, G is the complete graph over $V = Q \cup S$, and the weight of each edge (p, q) in G is the Euclidean distance between p and q. Arora [4] showed that the geometric Steiner tree problem can be efficiently approximated close to optimal.

A Steiner tree is *full* if all terminals are leaves of the tree. In the bottleneck full Steiner tree problem (BFST), the goal is to compute a full Steiner tree with minimum bottleneck (i.e., the length of the longest edge). The k-BFST problem is a restricted version of the BFST problem, for which, in addition to the sets Q and S, we are given a positive integer k, and the goal is to compute a full Steiner tree T with at most k Steiner points such that the bottleneck of T is minimized. Abu-Affash [1] gave a $O((n+m)\log^2 m)$ algorithm for the BFST problem and showed that the k-BFST problem is NP-hard but admits a polynomial-time 4-approximation algorithm. Later, Biniaz et al [10] gave an $O((n+m)\log m)$ algorithm for the BFST problem.

We consider the BFST and the k-BFST problems in bichromatic point sets. Given two sets of points in the plane; a set Q of n red and blue terminals and a set S of m red and blue Steiner points, the goal in the bottleneck bichromatic full Steiner tree (BBFST) problem is to find a full Steiner tree T such that each edge in T connects a red and a blue point and the bottleneck of T is minimized. We refer to this tree as a bichromatic full Steiner tree. In the k-BBFST problem, the goal is to compute a bichromatic full Steiner tree T with at most k Steiner points, such that its bottleneck is minimized, where $k \leq m$ is a given positive integer. The bichromatic input appeared in many geometric problems; for example, red-blue intersection [3], red-blue separation [5, 11, 13], and red-blue connection problems [2, 7].

In this paper, we show how to generalize the algorithms in [1] to solve the BBFST problem and to approximate the k-BBFST problem.

2 Exact Algorithm for BBFST

Given a set Q of n red and blue terminals and a set S of m red and blue Steiner points in the plane, we present an $O((n+m)\log^2 m)$ time algorithm that computes a bichromatic full Steiner tree of minimum bottleneck. We refer to such a tree as an optimal bichromatic full Steiner tree of Q.

Let Q_R and Q_B be the sets of red and blue terminal points of Q, respectively. Similarly, let S_R and S_B be the sets of red and blue Steiner points of S, respectively. Let $MST(S)$ be a minimum-weight bichro-

*Software Engineering Department, Shamoon College of Engineering, Beer-Sheva 84100, Israel, abuaa1@sce.ac.il.

†Department of Computer Science, Ben-Gurion University, Beer-Sheva 84105, Israel, sujoy.bhore@gmail.com. The research is partially supported by the Lynn and William Frankel Center for Computer Science.

‡Department of Computer Science, Ben-Gurion University, Beer-Sheva 84105, Israel, carmip@cs.bgu.ac.il. The research is partially supported by the Lynn and William Frankel Center for Computer Science.

§Advanced Computing and Microelectronics Unit, Indian Statistical Institute, Kolkata, India, dibyayancg@gmail.com.

matic spanning tree of S (i.e., of the complete bipartite graph of S_R and S_B). Let $S(T)$ be the set of Steiner points in a bichromatic full Steiner tree T.

Lemma 1 *There exists an optimal bichromatic full Steiner tree T^* of Q, such that $MST(S(T^*))$ is a subtree of $MST(S)$.*

Proof. Let T be an optimal bichromatic full Steiner tree of Q. Let $e = (p_r, p_b)$ be an edge in $MST(S(T))$ but not in $MST(S)$. Let P be the path between p_r and p_b in $MST(S)$. Since each edge in P is of length at most $|p_r p_b|$, $(T \setminus \{e\}) \cup P$ is also an optimal bichromatic full Steiner tree. By repeating this process for each edge $e \in MST(S(T)) \setminus MST(S)$, we obtain an optimal bichromatic full Steiner tree T^* satisfying the lemma. □

Let $e_1, e_2, \ldots, e_{m-1}$ be the edges of $MST(S)$ sorted in non-decreasing order by their length. For an edge $e_i \in MST(S)$, let $\mathcal{T}_i$ be the forest obtained from $MST(S)$ by deleting all edges of length greater than $|e_i|$ from $MST(S)$. By Lemma 1, there exists an optimal bichromatic full Steiner tree T^* of Q such that $MST(S(T^*))$ is a tree of $\mathcal{T}_i$, for some edge $e_i \in MST(S)$. Thus, by performing binary search on the lengths of edges of $MST(S)$, we can find a forest $\mathcal{T}_i$ that contains a tree T, such that, by connecting each point in Q to its closest point of opposite color in T, we obtain an optimal bichromatic full Steiner tree of Q.

For a point $p \in Q$ and a tree T, let $d(p, T)$ denote the distance between p and its closest point of opposite color in T. For an edge $e_i \in MST(S)$, to decide whether $\mathcal{T}_i$ contains a tree T, such that $d(q, T) \leq |e_i|$ for all $q \in Q$, can be implemented in $O((n+m)\log m)$ time [1, 9]. (In order to handle the case that the bottleneck of an optimal bichromatic full Steiner tree might be either less than $|e_1|$ or greater than $|e_{m-1}|$, we add the values $|e_0| = 0$ and $|e_m| = \infty$ to the search space.)

Let λ be the bottleneck of the optimal bichromatic full Steiner tree. Therefore, we can find an $0 \leq i \leq m-1$, such that $|e_i| < \lambda < |e_{i+1}|$ in $O((n+m)\log^2 m)$ time. If $|e_i| < \lambda < |e_{i+1}|$, then the optimal bichromatic full Steiner tree of Q is obtained by a tree T from the forest $\mathcal{T}_i$; see Figure 1(a). If $\lambda = |e_{i+1}|$, then the optimal bichromatic full Steiner tree of Q is obtained by a tree T from the forest $\mathcal{T}_{i+1}$; see Figure 1(b). Thus, in both cases, we can find the tree T in the set $\mathcal{T}_i \cup \mathcal{T}_{i+1}$, such that, by connecting each terminal in Q to its closest point of opposite color in T , we obtain an optimal bichromatic full Steiner tree of Q. We conclude by the following theorem.

Theorem 2 *The BBFST problem can be solved in $O((n+m)\log^2 m)$ time.*

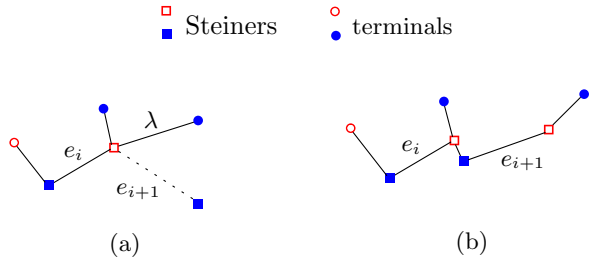


Figure 1: The optimal full bichromatic Steiner tree is obtained (a) from $\mathcal{T}_i$, when $|e_i| < \lambda < |e_{i+1}|$ and (b) from $\mathcal{T}_{i+1}$, when $\lambda = |e_{i+1}|$.

3 Approximation Algorithm for k-BBFST

Given two sets of points in the plane; a set Q of n red and blue terminal points, a set S of m red and blue Steiner points, and a positive integer $k \leq m$, the goal in the k-BBFST problem is to compute a bichromatic full Steiner tree with at most k Steiner points from S and its bottleneck is minimized. In this section, we first prove that the k-BBFST problem is NP-hard. Then, we present a polynomial-time approximation algorithm with performance ratio 9.

3.1 Hardness proof

Due to space limitation the proof of the following theorem has moved to the appendix.

Theorem 3 *The k-BBFST problem is NP-hard.*

3.2 Approximation algorithm

We devise a polynomial-time approximation algorithm for computing a bichromatic full Steiner tree with at most k Steiner points (k-BFST for short), such that its bottleneck is at most 9 times the bottleneck of an optimal k-BFST.

Let Q_R and Q_B be the sets of red and blue terminal points of Q, respectively. Similarly, let S_R and S_B be the sets of red and blue Steiner points of S, respectively. Let $G = (V, E)$ be the graph with $V = Q \cup S$ and $E = (Q_R \times S_B) \cup (Q_B \times S_R) \cup (S_R \times S_B)$. We assume, w.l.o.g., that $E = \{e_1, e_2, \cdots, e_l\}$, such that $|e_1| \leq |e_2| \leq \cdots \leq |e_l|$. Notice that the bottleneck of an optimal k-BFST is a length of an edge from E. For an edge e_i, Let $G_i = (V, E_i)$ be the graph, such that $E_i = \{e_j \in E : |e_j| \leq |e_i|\}$. We devise a procedure which either constructs a k-BFST of Q in G with bottleneck at most 9 times $|e_i|$ or it says that G_i does not contain a k-BFST of Q.

Let G_i^2 be the 2nd power graph of G_i, i.e., G_i^2 has the same set of vertices as G_i and an edge between two vertices if and only if there is a path that contains at most 2 edges between them in G_i. Let $G_i^2(Q)$ be the sub-graph of G_i^2 induced by Q and let Q' be a maximal

independent set in $G_i^2(Q)$. Notice that, since all the edges in E are bichromatic, a red terminal and a blue terminal cannot be connected to a same Steiner point in G_i. Hence, a red terminal and a blue terminal cannot be connected to each other in G_i^2. Thus, if $|Q'| = 1$, then Q contains points of one color and we can construct a k-BFST of bottleneck at most $3|e_i|$ as follows. Let p be the only point in Q' and assume, w.l.o.g., that p is a red point. We select a blue Steiner point s that is connected to p in G_i and we connect it to all points of Q. Since there is an edge in G_i^2 between p and each other point $q \in Q$, we have $|pq| \leq 2|e_i|$, and therefore, $|sq| \leq 3|e_i|$.

Thus, we assume that $|Q'| > 1$. For any two points $p, q \in Q$, let $\delta_i(p, q)$ be the path between p and q in G_i that contains minimum number of Steiner points. Let $G' = (Q', E')$ be the complete graph over Q'. For each edge (p, q) in E', we assign a weight $w(p, q)$ which is equal to the number of Steiner points in $\delta_i(p, q)$. Let $MST(G')$ be the minimum spanning tree of G' under w. We define the normalized weight of $MST(G')$ as $W(MST(G')) = \sum_{e \in MST(G')} \lfloor w(e)/2 \rfloor$.

Lemma 4 *If G_i contains a k-BFST of Q', then $W(MST(G')) \leq k$*

Proof. Let T be a k-BFST of Q' in G_i. We construct a spanning tree T' of G' such that $W(T') \leq k$. We start by T and we transform it into T' by an iterative process. We start by selecting an arbitrary Steiner point as the root of T; see Figure 2. In each iteration, we select the deepest leaf p in the rooted tree, which is a terminal, and we connect it to its nearest terminal q by an edge (p, q) of weight equal to the number of Steiner points between them. Let s be the first common ancestor of p and q. We then remove the Steiner points between p and s. In the last iteration, we remove all of the remaining points.

Since, in each iteration, we select the deepest terminal, we add to T' an edge (p, q) of weight $w(p, q)$, and we remove at least $\lfloor w(p, q)/2 \rfloor$ Steiner points from T. Thus, we have $W(T') = \sum_{e \in T'} \lfloor w(e)/2 \rfloor \leq k$. Finally, since T' is also a spanning tree of G', we have $W(MST(G')) \leq W(T') \leq k$. □

We now describe the algorithm. For each edge $e_i \in E$ in the sorted order, we construct the graphs G_i, G_i^2, and $G_i^2(Q)$. Then, we compute a maximal independent set Q' in $G_i^2(Q)$. If $|Q'| = 1$, then we construct a k-BFST of Q with bottleneck at most 3 times $|e_i|$. Otherwise, we construct the complete graph G' over Q', and we compute a minimum spanning tree $MST(G')$ of G' with respect to the weight function w. If $W(MST(G')) > k$, then we proceed to the next edge e_{i+1}. Otherwise, we construct a k-BFST of Q with bottleneck at most 9 times $|e_i|$ as follows.

For each edge $(p, q) \in T$, there is a bichromatic path $\delta_i(p, q)$ between p and q in G_i that contains $w(p, q)$ Steiner points. We select $\lfloor w(p, q)/2 \rfloor$ Steiner points on any shortest Steiner path between p and q in G_i by the following procedure.

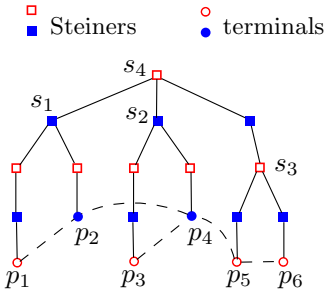


Figure 2: Constructing T' from T. In iteration 1, we select p_1, connect it to p_2 by an edge of weight 4 and remove the points between p_1 and s_1. In iteration 2, we select p_3, connect it to p_4 by an edge of weight 4, and remove the points between p_3 and s_2. In iteration 3, we select p_6, connect it to p_5 by an edge of weight 3, and remove the points between p_6 and s_3. In iteration 4, we select p_5, connect it to p_4 by an edge of weight 6, and remove the points between p_5 and s_4. In the last iteration, we select p_2, connect it to p_4 by an edge of weight 5, and remove the all the remaining points between p_2 and p_4.

We select an arbitrary leaf p in $MST(G')$ and we traverse $MST(G')$ starting from p. Let q be the point that is connected to p in $MST(G')$. Set $S' = \emptyset$. We call the recursive procedure $SelectSteiners(p, q, color(p), S')$ (Procedure 1) that selects at most k Steiner points and adds them to S'; see also Figure 5 (in the appendix).

Procedure 1 $SelectSteiners(p, q, color, S')$

```
1: j ← w(p, q)
2: let s_1, s_2, ..., s_j be the Steiner points in δ_i(p, q)
3: x ← 0
4: if color(s_1) ≠ color then
5:     i ← 1
6: else
7:     i ← 2
8: while i + 3x ≤ j do
       S' ← S' ∪ {s_{i+3x}}
       x ← x + 1
9: for each (q, t) ∈ MST(G'), such that t ≠ p do
       SelectSteiners(q, t, color(s_{i+3(x−1)}), S')
```

It is not hard to see that for each edge (p, q) in $MST(G')$, we add to S' at most $\lfloor w(p, q)/2 \rfloor$ Steiner points. Therefore, $|S'| \leq k$. Next, we construct a minimum spanning tree $MST(S')$ of S' (i.e., of the complete Euclidean graph over S'). Notice that, each edge in $MST(S')$ is of length at most $5|e_i|$; see Figure 5 (in the appendix). Finally, we connect each terminal in Q to its nearest opposite color Steiner point

in S' to obtain a bichromatic full Steiner tree. This guarantees that each terminal in Q' is connected to a Steiner point with an edge of length at most $7|e_i|$; see Figure 5 (in the appendix), and each terminal in $Q \setminus Q'$ is connected to a Steiner point with an edge of length at most $9|e_i|$.

Remark. If Q' contains only one red and one blue points p and q, respectively, $k = 2$, and $MST(G')$ is a path between p and q that contains exactly 2 Steiner points, a blue Steiner point s_1 and a red Steiner point s_2, then we construct a k-BFST by connecting all the points in Q_R to s_1 and all the points in Q_B to s_2. This k-BFST contains exactly 2 Steiner points and its bottleneck is at most $3|e_i|$.

Lemma 5 *Our algorithm constructs a k-BFST of Q with bottleneck at most 9 times the bottleneck of an optimal k-BFST.*

Proof. Let $e_i \in E$ be the first edge satisfying $W(T) \leq k$. Thus, by Lemma 4, the bottleneck of any k-BFST in G is at least $|e_i|$. Therefore, the constructed k-BFST has a bottleneck at most 9 times the bottleneck of an optimal k-BFST. □

Lemma 6 *Our algorithm can be implemented in polynomial time.*

Due to space limitation the proof of this lemma is moved to the appendix.

Theorem 7 *The above algorithm computes a k-BFST with bottleneck at most 9 times the bottleneck of an optimal k-BFST in polynomial time.*

References

[1] A. K. Abu-Affash. The Euclidean bottleneck full steiner tree problem. *Algorithmica*, 71(1):139–151, 2015.

[2] P. K. Agarwal, H. Edelsbrunner, and O. Schwarzkopf. Euclidean minimum spanning trees and bichromatic closest pairs. *Disc. & Comput. Geom.*, 6:407–422, 1991.

[3] P. K. Agarwal and M. Sharir. Red-blue intersection detection algorithms, with applications to motion planning and collision detection. *SIAM J. Comput.*, 19(2):297–321, 1990.

[4] S. Arora. Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. *J. ACM*, 45(5):753–782, 1998.

[5] S. Arora and K. L. Chang. Approximation schemes for degree-restricted MST and red-blue separation problem. In *Proceedings of ICALP*, pages 176–188, 2003.

[6] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, 1998.

[7] M. J. Atallah and D. Z. Chen. On connecting red and blue rectilinear polygonal obstacles with nonintersecting monotone rectilinear paths. *Int. J. Comput. Geom. Appl.*, 11(4):373–400, 2001.

[8] P. Berman and V. Ramaiyer. Improved approximations for the steiner tree problem. In *Proceedings of SODA*, pages 325–334, 1992.

[9] A. Biniaz, P. Bose, D. Eppstein, A. Maheshwari, P. Morin, and M. Smid. Spanning Trees in Multipartite Geometric Graphs. *ArXiv e-prints*, nov 2016.

[10] A. Biniaz, A. Maheshwari, and M. Smid. An optimal algorithm for the Euclidean bottleneck full steiner tree problem. *Comput. Geom.*, 47(3):377–380, 2014.

[11] J.-D. Boissonnat, J. Czyzowicz, O. Devillers, J. Urrutia, and M. Yvinec. Computing largest circles separating two sets of segments. *Int. J. Comput. Geom. Appl.*, 10(1):41–53, 2000.

[12] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms, 3rd edition.* The MIT Press, 2009.

[13] E. D. Demaine, J. Erickson, F. Hurtado, J. Iacono, S. Langerman, H. Meijer, M. H. Overmars, and S. Whitesides. Separating point sets in polygonal environments. *Int. J. Comput. Geom. Appl.*, 15(4):403–420, 2005.

[14] M. R. Garey, R. L. Graham, and D. S. Johnson. The complexity of computing steiner minimal trees. *SIAM J. Appl. Math.*, 32(4):835–859, 1977.

[15] M. R. Garey and D. S. Johnson. The rectilinear Steiner tree problem is NP-complete. *SIAM J. Appl. Math.*, 32(4):826–834, 1977.

[16] M. Karpinski and A. Zelikovsky. New approximation algorithms for the steiner tree problems. *J. Comb. Opt.*, 1(1):47–65, 1997.

[17] H. J. Prömel and A. Steger. A new approximation algorithm for the steiner tree problem with performance ratio 5/3. *Journal of Algorithms*, 36(1):89–101, 2000.

[18] W. Schnyder. Embedding planar graphs on the grid. In *Proceedings of SODA*, pages 138–148, 1990.

Computing the Geometric Intersection Number of Curves*

Vincent Despré, Francis Lazarus

Abstract

The geometric intersection number of a curve on a surface is the minimal number of self-intersections of any homotopic curve, i.e. of any curve obtained by continuous deformation. Given a curve c represented by a closed walk of length at most ℓ on a combinatorial surface of complexity n we describe simple algorithms to (1) compute the geometric intersection number of c in $O(n+\ell^2)$ time, (2) construct a curve homotopic to c that realizes this geometric intersection number in $O(n+\ell^4)$ time, (3) decide if the geometric intersection number of c is zero, i.e. if c is homotopic to a simple curve, in $O(n+\ell\log^2\ell)$ time.

To our knowledge, no exact complexity analysis had yet appeared on those problems. An optimistic analysis of the complexity of the published algorithms for problems (1) and (3) gives at best a $O(n+g\ell^2)$ time complexity on a genus g surface without boundary. No polynomial time algorithm was known for problem (2). Our solution to problem (3) is the first quasi-linear algorithm since the problem was raised by Poincaré more than a century ago. Finally, we note that our algorithm for problem (1) extends to compute the geometric intersection number of two curves of length at most ℓ in $O(n+\ell^2)$ time.

1 Introduction

Let S be a surface. Two closed curves $\alpha,\beta:\mathbb{R}/\mathbb{Z}\to S$ are **freely homotopic**, written $\alpha\sim\beta$, if there exists a continuous map $h:[0,1]\times\mathbb{R}/\mathbb{Z}$ such that $h(0,t)=\alpha(t)$ and $h(1,t)=\beta(t)$ for all $t\in\mathbb{R}/\mathbb{Z}$. Assuming the curves are in general position, their number of intersections is

$$|\alpha\cap\beta| = |\{(t,t') \mid t,t'\in\mathbb{R}/\mathbb{Z} \text{ and } \alpha(t)=\beta(t')\}|.$$

Their **geometric intersection number** only depends on their free homotopy classes and is defined as

$$i(\alpha,\beta) = \min_{\alpha'\sim\alpha,\beta'\sim\beta} |\alpha'\cap\beta'|.$$

Likewise, the number of self-intersections of α is given by

$$\frac{1}{2}|\{(t,t') \mid t\neq t'\in\mathbb{R}/\mathbb{Z} \text{ and } \alpha(t)=\alpha(t')\}|,$$

and its minimum over all the curves freely homotopic to α is its **geometric self-intersection number** $i(\alpha)$. Note the one half factor that comes from the identification of (t,t') with (t',t).

The geometric intersection number is an important parameter that allows to stratify the set of homotopy classes of curves on a surface. The surface is usually endowed with a hyperbolic metric, implying that each homotopy class is identified by its unique geodesic representative. Extending a former result by Mirzakhani [9], Sapir [12, 10] has recently provided bounds for the number of closed geodesics with bounded length and bounded geometric intersection number. Other more experimental results were obtained with the help of a computer to show the existence of (word-)length-equivalent homotopy classes with distinct geometric intersection numbers [3]. Hence, for both theoretical and practical reasons various aspects of the computation of geometric intersection numbers have been studied in the past including the algorithmic ones. Nonetheless, all the previous approaches rely on rather complex mathematical arguments and to our knowledge no exact complexity analysis has yet appeared. In this paper, we make our own the words of Dehn who noted that the metric on words (on some basis of the fundamental group of the surface) can advantageously replace the hyperbolic metric. We propose a combinatorial framework that leads to simple algorithms of low complexity to compute the geometric intersection number of curves or to test if this number is zero. Our approach is based on the computation of canonical forms as recently introduced in the purpose of testing whether two curves are homotopic [7, 5]. Canonical forms are instances of combinatorial geodesics who share nice properties with the geodesics of a hyperbolic surface. On such surfaces each homotopy class contains a unique geodesic that moreover minimizes the number of self-intersections. Although a combinatorial geodesic is generally not unique in its homotopy class, it must stay at distance one from its canonical representative and a careful analysis of its structure leads to the first result of the paper.

Theorem 1 *Given two curves represented by closed walks of length at most ℓ on a combinatorial surface of complexity n we can compute the geometric intersection number of each curve or of the two curves in $O(n+\ell^2)$ time.*

As usual the complexity of a combinatorial surface

*This work was partially supported by the LabEx PERSYVAL-Lab ANR-11-LABX-0025-01.

stands for its total number of vertices, edges and faces. A key point in our algorithm is the ability to compute the primitive root of a canonical curve c in linear time. This is a curve r that is not homotopic to a proper power of any other curve and such that $c \sim r^k$ for some integer k. We next provide an algorithm to compute an actual curve immersion that minimizes the number of self-intersections in its homotopy class.

Theorem 2 *Let c be a closed walk of length ℓ in canonical form. We can compute a combinatorial immersion with $i(c)$ crossings in $O(\ell^4)$ time.*

We also propose a nearly optimal algorithm that answers an old problem studied by Poincaré [11, §4]: decide if the geometric intersection number of a curve is null, that is if the curve is homotopic to a simple curve.

Theorem 3 *Given a curve represented by a closed walk of length ℓ on a combinatorial surface of complexity n we can decide if the curve is homotopic to a simple curve in $O(n + \ell \log^2 \ell)$ time.*

We emphasize that our results represent significant progress with respect to the state of the art. Even though no precise analysis appeared in the previously proposed algorithms [2, 8, 4] concerning Theorems 1 or 3, an optimistic analysis of what seems the most efficient approach [8] gives at best a quadratic time complexity for computing the geometric number on a genus g surface without boundary, assuming that the curves are primitive. Apart from a recent algorithm by Aretinnes [1], which is restricted to surfaces with nonempty boundary, we know of no polynomial time algorithm for Theorem 2. Finally, Theorem 3 states the first quasi-linear algorithm for detecting homotopy classes of simple curves since the problem was raised by Poincaré more than a century ago [11, §4].

Due to page limitation we will focus on Theorem 1. In Section 2 we review the relevant definitions and background. In Section 3, we study geodesics in systems of quads, a particular kind of combinatorial surfaces that we use to compute the geometric intersection number. Section 4 presents our strategy for counting intersections. The end of the proof of Theorem 1 involves some technical details, we decide to only show how we handle the non-primitive case in Section 5. The missing details and the proofs of the two other theorems can be found in the full version of the paper `https://arxiv.org/pdf/1511.09327.pdf`.

2 Background and framework

Combinatorial surfaces. As usual in computational topology, we model a surface by a cellular embedding of a graph G in a compact topological surface S. Such a cellular embedding can be encoded by a **combinatorial surface** composed of the graph G itself together with a rotation system that records for every vertex of the graph the clockwise cyclic order of the incident arcs. The **facial walks** are obtained from the rotation system by the face traversal procedure. All the considered graphs may have loop and multiple edges. A directed edge will be called an **arc** and each edge corresponds to two opposite arcs. We denote by a^{-1} the arc opposite to an arc a. For simplicity, **we shall only consider orientable surfaces in this paper**. Every combinatorial surface Σ can be reduced by first contracting the edges of a spanning tree and then deleting edges incident to distinct faces. The resulting **reduced surface** has a single vertex and a single face. The combinatorial surface Σ and its reduced version encode different cellular embeddings on a same topological surface.

Combinatorial curves. Consider a combinatorial surface with its graph G. A combinatorial curve (or path) c is a walk in G, i.e. an alternating sequence of vertices and arcs, starting and ending with a vertex, such that each vertex in the sequence is the target vertex of the previous arc and the source vertex of the next arc. We generally omit the vertices in the sequence. A combinatorial curve is closed when additionally the first and last vertex are equal. When no confusion is possible we shall drop the adjective combinatorial. A **spur** of c is a subsequence of arcs of the form (a, a^{-1}). A closed curve is **contractible** if it is homotopic to a trivial curve (i.e., a curve reduced to a single vertex). We will implicitly assume that a homotopy has fixed endpoints when applied to paths and is free when applied to closed curves.

Diagrams. A **disk diagram** over the combinatorial surface Σ is a combinatorial disk Δ together with a labelling of the arcs of Δ by the arcs of Σ such that opposite arcs receive opposite labels, and the facial walk of each non-perforated face of Δ is labelled by the facial walk of some non-perforated face of Σ. An **annular diagram** is defined the same way with a combinatorial annulus instead of a combinatorial disk.

Reduction to a system of quads. For the rest of the paper we shall assume that **all surfaces have negative Euler characteristic**. Let Σ be a combinatorial surface with negative Euler characteristic. Following Lazarus and Rivaud [7] we start putting Σ into a standard form called a **system of quads** by Erickson and Whittlesey [5]. After reducing Σ to a surface Σ' with a single vertex v and a single face f this system of quads is obtained by adding a vertex w at the center of f, adding edges between w and all occurrences of v in the facial walk of f, and finally deleting the edges of Σ'. The graph of the resulting

system of quads, called the **radial graph** [7], is bipartite. It contains two vertices, namely v and w, and $4g$ edges, where g is the genus of Σ. All its faces are quadrilaterals. Our initial problem can be carried into a system of quads using those techniques. Thus, wee may assume that our combinatorial surface Σ is a system of quads.

Spurs, brackets and canonical curves. Following the terminology of Erickson and Whittlesey [5], we define the **turn** of a pair of arcs (a_1, a_2) sharing their origin vertex v as the number of face corners between a_1 and a_2 in clockwise order around v. Hence, if v is a vertex of degree d in Σ, the turn of (a_1, a_2) is an integer modulo d that is zero when $a_1 = a_2$. The **turn sequence** of a subpath $(a_i, a_{i+1}, \ldots, a_{i+j-1})$ of a closed curve of length ℓ is the sequence of $j+1$ turns of $(a_{i+k}^{-1}, a_{i+k+1})$ for $-1 \leq k < j$, where indices are taken modulo ℓ. The subpath may have length ℓ, thus leading to a sequence of $\ell+1$ turns. Note that the turn of $(a_{i+k}^{-1}, a_{i+k+1})$ is zero precisely when (a_{i+k}, a_{i+k+1}) is a spur. A **bracket** is any subpath whose turn sequence has the form 12^*1 or $\bar{1}\bar{2}^*\bar{1}$, where t^* stands for a possibly empty sequence of turns t and $\bar{x}$ stands for $-x$. Lazarus and Rivaud [7] have introduced a canonical form for every free homotopy class of closed curves in a system of quads. In particular, two curves are freely homotopic if and only if their **canonical forms** are equal. It was further characterized by Erickson and Whittlesey [5] in terms of turns and brackets. It is the unique homotopic curve that contains no spurs or brackets and whose turning sequence contains no -1's and contains at least one turn that is not -2.

Theorem 4 ([7, 5]) *The canonical form of a closed curve of length ℓ on a system of quads can be computed in $O(\ell)$ time.*

3 Geodesics

The canonical form is an instance of a **combinatorial geodesic**, i.e. a curve that contains no spurs or brackets. The canonical form is the rightmost homotopic geodesic. The definitions of a geodesic and of a canonical form extend trivially to paths. Although we cannot claim in general the uniqueness of geodesics in a homotopy class, homotopic geodesics are almost equal and have the same length. Specifically, define a **(quad) staircase** as a planar sequence of quads obtained by stitching an alternating sequence of rows and columns of quads to form a staircase. Assuming that the staircase goes up from left to right, we define the **initial tip** of a quad staircase as the lower left vertex of the first quad in the sequence. The **final tip** is defined as the upper right vertex of the last quad. See Fig. 1. A **closed staircase** is obtained by identifying the two vertical arcs incident to the initial and final tips of a staircase.

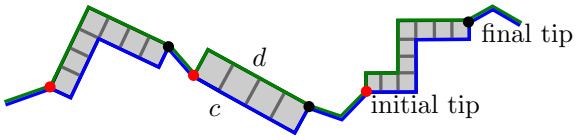


Figure 1: A disk diagram for two homotopic paths c and d composed of paths and staircases.

Theorem 5 *Let c, d be two non-trivial homotopic combinatorial geodesics. If c, d are closed curves, then they label the two boundary cycles of an annular diagram composed of a unique closed staircase or of an alternating sequence of paths (possibly reduced to a vertex) and quad staircases connected through their tips. Likewise, if c, d are paths, then the closed curve $c \cdot d^{-1}$ labels the boundary of a disk diagram composed of an alternating sequence of paths (possibly reduced to a vertex) and quad staircases connected through their tips.*

Corollary 6 *With the hypothesis of Theorem 5, c and d have equal length which is minimal among homotopic curves. Moreover, c and d have no contractible subpath.*

The next remark follows directly from the characterization of geodesics and canonical forms in terms of spurs, brackets and turns.

Remark 1 *Likewise, any power c^k of a combinatorial closed geodesic c is also a combinatorial geodesic. Moreover, if c is in canonical form, so is c^k.*

4 Our strategy for counting intersections

Following Poincaré's original approach we represent the surface S as the hyperbolic quotient surface $\mathbb{D}/\Gamma$ where Γ is a discrete group of hyperbolic motions of the Poincaré disk $\mathbb{D}$. We denote by $p : \mathbb{D} \to \mathbb{D}/\Gamma = S$ its universal covering map. Any closed curve $\alpha : \mathbb{R}/\mathbb{Z} \to S$ gives rise to its infinite power $\alpha^\infty : \mathbb{R} \to \mathbb{R}/\mathbb{Z} \to S$ that wraps around α infinitely many times. A **lift** of α is any curve $\tilde{\alpha} : \mathbb{R} \to \mathbb{D}$ such that $p \circ \tilde{\alpha} = \alpha^\infty$ where the parameter of $\tilde{\alpha}$ is defined up to an integer translation (we thus identify the curves $t \mapsto \tilde{\alpha}(t+k)$, $k \in \mathbb{Z}$). Note that $p^{-1}(\alpha)$ is the union of all the images $\Gamma \cdot \tilde{\alpha}$ of $\tilde{\alpha}$ by the motions in Γ. The curve $\tilde{\alpha}$ has two limit points on the boundary of $\mathbb{D}$ which can be joined by a unique hyperbolic line L. The projection $p(L)$ covers infinitely many times around the unique geodesic homotopic to α. In particular, the limit points of $\tilde{\alpha}$ are independent of the chosen representative in the homotopy class of α.

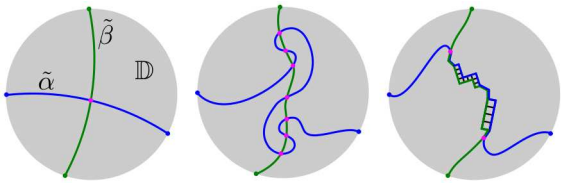


Figure 2: Left, two intersecting hyperbolic lines. Middle, two lifts of non-geodesic curves may intersect several times. Right, lifts of combinatorial geodesics.

No two motions of Γ have a limit point in common unless they are powers of the same motion. This can be used to show that when α is primitive its lifts are uniquely identified by their limit points. Let α and β be two primitive curves. We fix a lift $\tilde{\alpha}$ of α and denote by $\tau \in \Gamma$ the hyperbolic motion sending $\tilde{\alpha}(0)$ to $\tilde{\alpha}(1)$. Let $\Gamma \cdot \tilde{\beta}$ be the set of lifts of β. We consider the subset of lifts

$$B = \{\tilde{\beta}' \in \Gamma \cdot \tilde{\beta} \mid \text{ the limit points of } \tilde{\beta}' \text{ and } \tilde{\alpha} \text{ alternate along } \partial\mathbb{D}\},$$

and we denote by B/τ the set of equivalence classes of lifts generated by the relations $\tilde{\beta}' \sim \tau(\tilde{\beta}')$.

Lemma 7 *$i(\alpha, \beta) = |B/\tau|$.*

In the ideal situation of hyperbolic geodesics, each intersection point of α and β corresponds precisely to a class in B/τ. When α and β are not geodesic the situation is more ambiguous and their lifts may have multiple intersection points. When dealing with combinatorial geodesics, the situation is more constrained and somehow intermediate between the hyperbolic case and the most general situation. See Figure 2. Our strategy to compute the geometric intersection number consists of identifying B/τ with certain pairs of homotopic subpaths of α and β in our combinatorial framework.

5 Non-primitive curves 1

Thanks to canonical forms, computing the primitive root of a curve becomes extremely simple.

Lemma 8 *Let c be a combinatorial curve of length $\ell > 0$ in canonical form. A primitive curve d such that c is homotopic to d^k for some integer k can be computed in $O(\ell)$ time.*

Proof. By Theorem 4, we may assume that c and d are in canonical form. By Remark 1, the curve d^k is also in canonical form. The uniqueness of the canonical form implies that $c = d^k$, possibly after some circular shift of d. It follows that d is the smallest prefix of c such that c is a power of this prefix. It can be found in $O(\ell)$ time using a variation of the Knuth-Morris-Pratt algorithm to find the smallest period of a word [6]. □

The geometric intersection number of non-primitive curves is related to the geometric intersection number of their primitive roots. The next result is part of the folklore.

Proposition 9 ([4]) *Let c and d be primitive curves and let p, q be positive integers. Then,*

$$i(c^p) = p^2 \times i(c) + p - 1 \quad \text{and}$$

$$i(c^p, d^q) = \begin{cases} 2pq \times i(c) & \text{if } c \sim d \text{ or } c \sim d^{-1}, \\ pq \times i(c, d) & \text{otherwise.} \end{cases}$$

Those two results justify the restriction to primitive curves in the previous sections.

References

[1] C. Arettines. A combinatorial algorithm for visualizing representatives with minimal self-intersection. *J. Knot Theor. Ramif.*, 2015.

[2] J. S. Birman and C. Series. An algorithm for simple curves on surfaces. *J. London Math. Soc.*, 29(2):331–342, 1984.

[3] M. Chas. Self-intersection numbers of length-equivalent curves on surfaces. *Exp. Math.*, 23(3):271–276, 2014.

[4] M. de Graaf and A. Schrijver. Making curves minimally crossing by Reidemeister moves. *J. Com. Theory B*, 70(1):134–156, 1997.

[5] J. Erickson and K. Whittelsey. Transforming curves on surfaces redux. In *Proc. 24th ACM-SIAM Symp. Discrete Alg. (SODA)*, 2013.

[6] D. E. Knuth, J. H. Morris, Jr, and V. R. Pratt. Fast pattern matching in strings. *SIAM J. Comput.*, 6(2):323–350, 1977.

[7] F. Lazarus and J. Rivaud. On the homotopy test on surfaces. In *Proc. 53rd IEEE Symp. Found. Comput. Sci. (FOCS)*, pages 440–449, 2012.

[8] M. Lustig. Paths of geodesics and geometric intersection numbers: II. In *Combinatorial group theory and topology*, volume 111 of *Ann. of Math. Stud.*, pages 501–543. Princeton Univ. Press, 1987.

[9] M. Mirzakhani. Growth of the number of simple closed geodesies on hyperbolic surfaces. *Ann. Math.*, pages 97–125, 2008.

[10] M. Mirzakhani. Counting mapping class group orbits on hyperbolic surfaces. Preprint, 2016.

[11] Henri Poincaré. Cinquième complément à l'analysis situs. *Rendiconti del Circolo Matematico di Palermo*, 18(1):45 –110, 1904.

[12] J. Sapir. Bounds on the number of non-simple closed geodesics on a surface. Preprint, 2015.

K-Dominance in Multidimensional Data

Thomas Schibler* Subhash Suri†

Abstract

We derive bounds on the number of k-dominant maxima points in d dimensions, and propose an efficient algorithm for computing them.

1 Introduction

Given a finite set of points $\mathcal{V}$ in R^d, a point u is said to *k-dominate* another point v if $u_i \geq v_i$ holds for k of the dimensions, $i = 1, 2, \ldots, d$, with strict inequality in at least one dimension. We use the notation $u \succ_k v$ to indicate k-domination of v by u, and write $u \not\succ_k v$ when u does not k-dominate v. The *k-dominant skyline* of $\mathcal{V}$, denoted $\mathcal{KDS}(\mathcal{V}, k)$, is the set of points that are not k-dominated:

$$\mathcal{KDS}(\mathcal{V}, k) \;=\; \{v \in \mathcal{V} \mid u \;\not\succ_k\; v,\; \forall\, u \in \mathcal{V} \setminus \{v\}\},$$

The k-dominant skyline ($\mathcal{KDS}$) is a generalization of the skyline, also called the maxima points, which is the subset of points not dominated on all d coordinates by any other point in the input [1, 2, 3]. The $\mathcal{KDS}$ was introduced recently as a way to deal with the problem that far too many points can appear on the skyline in high dimensions [5]. For instance, a database query for a car or a smart phone can easily produce an overwhelming number of *incomparable choices* (skyline points), with no obvious way to rank them. (Although in theory all input points can appear on the skyline even in two dimensions, this pathological behavior is rarely observed in low-dimensional real-world data.) By relaxing d-dominance to k-dominance, for $k < d$, many more points can be eliminated from the skyline, resulting in a smaller, more manageable, set of maxima points in higher dimensions.

1.1 Results

In this paper, we analyze the cardinality of the k-dominant skyline under both the average case, where the point coordinates are chosen independently from a distribution, and the worst-case, where the points are chosen adversarily. We also develop a sub-quadratic algorithm for computing $\mathcal{KDS}$.

1. Let $\mathcal{V}$ be a set of n points in d dimensions where the components of each point are distributed independently of each other, and for each component the magnitudes form a random permutation of $\{1, 2, \ldots, n\}$. Then, the expected number of points appearing on the k-dominant skyline is

$$\begin{cases} O\left((\log n)^{2k-d-1}\right) & \text{for } k > \frac{d+1}{2} \\ O(1) & \text{otherwise} \end{cases}$$

   Our result smoothly interpolates the cardinality of $\mathcal{KDS}$ for all values of k, and subsumes the classical result of Bentley et al. [2] as a special case for $k = d$.

2. The worst-case cardinality of $\mathcal{KDS}$, when input points are chosen adversarily, is n for all $k > (d+1)/2$, and 1 for all $k \leq (d+1)/2$.

3. We show that the $\mathcal{KDS}$ can be found by computing multiple (traditional) skylines of points projected onto k-dimensional spaces. Our algorithm runs in time $O\left(d^{d-k} n \log^{k-1} n\right)$, which is subquadratic for dimensions $d \leq c \log n / \log\log n$, for some constant c.

1.2 Related Work

The problem of computing or estimating the size of the skyline of multi-dimensional data has a long history in mathematics, computational geometry, and databases [1, 2, 3, 10, 12, 13, 14, 16, 9, 15], although the quest for efficient algorithms that scale to high dimensions and large input continues to this day. It has also been observed that as the dimension grows, the skylines can quickly lose their utility because most of the input points may appear on the skyline [5, 7], prompting the introduction of k-dominant skylines.

*Department of Computer Science, University of California, Santa Barbara, CA 93106, USA. tschibler@gmail.com

†Department of Computer Science, University of California, Santa Barbara, CA 93106, USA. suri@cs.ucsb.edu

The focus of our paper is to derive upper bounds (ceiling) on the size of the $\mathcal{KDS}$. There exists a substantial and rich literature on estimating the cardinality of (conventional) skylines [2, 9, 10, 14, 16], under a variety of data models, including in-memory, distributed, and data streams. The most relevant to our work is the classical result of Bentley et al [2], which proves that the expected size of the skyline of n vectors, whose components are chosen independently at random, is $O((\log n)^{d-1})$ in d-dimensions; the bound was later improved slightly to $O((\log n)^{d-1}/(d-1)!)$ by Buchta [4]. However, very little is known about the cardinality of the $\mathcal{KDS}$, which was introduced in [5] but the primary focus of that paper is heuristic methods for computing the skyline efficiently and their empirical evaluation. In a related work, Chan et al. [6] compute points that appear in many k-dominant skylines, but again the paper is concerned with the design and evaluation of an efficient heuristic. In [11], Hwang et al. consider certain threshold phenomena in k-dominant skylines under a continuous probability model, and derive limit bounds as $n, d \to \infty$. By contrast, our goal is to establish parametric upper bounds on the cardinality of $\mathcal{KDS}$ under a random model of input, similar to those for the conventional skylines obtained by Bentley et al. [2].

2 Cardinality of KDS

The worst-case bounds for $\mathcal{KDS}$ are easy to prove. In particular, we have the following result, whose proof is omitted due to lack of space.

Theorem 1 *The worst-case cardinality of $\mathcal{KDS}$ obeys the following bounds:*

1. *Given any set of n points in d-space and any k with $k \leq (d+1)/2$, $|\mathcal{KDS}(\mathcal{V}, k)| \leq 1$.*
2. *For any $n \geq 1$, and k, d such that $k > (d+1)/2$, there exists a set $\mathcal{V}$ of n points in d-space for which $|\mathcal{KDS}(\mathcal{V}, k)| = n$.*

The more interesting, and non-trivial, result concerns the average case behavior. Our analysis uses the standard "attribute distinctness and statistical independence" model [2, 9], which only assumes that the point components are distributed independently of each other, and for each component the magnitudes form a random permutation of $\{1, 2, \ldots, n\}$, namely, a total rank ordering. We interpret the input set of n points as an $n \times d$ array, whose rows are the points and whose columns are permutations of $\{1, 2, \ldots, n\}$. The set of all possible permutations produces exactly $(n!)^d$ distinct points. We analyze the complexity of the k-dominant skyline for an input array $\mathcal{V}$ chosen uniformly at random from this set. What is the expected size of the k-dominant skyline for such an input $\mathcal{V}$?

We analyze the average size of $\mathcal{KDS}$ by setting up a recurrence. In order to aid that derivation, let $A(n, d, k)$ denote the average size of the k-dominant skyline for a set of n points in d dimensions, where the points are chosen under the random model described above. We assume, without loss of generality, that the first column of the input $n \times d$ array is sorted in the ascending order $(1, 2, \ldots, n)$—that is, the first coordinate of the ith point is i, which if necessary can be realized by simply relabeling the points. See Figure 1 for illustration.

Let us focus on a single but arbitrary point v, and derive the probability that it belongs to the k-dominant skyline. Suppose the point v is represented as the ith row, which means its first coordinate is $v_1 = i$. We partition the remaining set of input points into two groups:

$$\begin{aligned} \mathcal{V}_a &= \{u \in \mathcal{V} \mid u_1 < v_1\} \\ \mathcal{V}_b &= \{u \in \mathcal{V} \mid u_1 > v_1\} \end{aligned}$$

That is, $\mathcal{V}_a$ is the set of points that v dominates on the first coordinate, and $\mathcal{V}_b$ is the set of points that dominate v on the first coordinate. The key observation is that for v to be a $\mathcal{KDS}$ point both of the following two events must occur:

1. None of the points in $\mathcal{V}_a$ k-dominates v among dimensions $\{2, 3, \ldots, d\}$, and
2. None of the points in $\mathcal{V}_b$ $(k-1)$-dominates v among dimensions $\{2, 3, \ldots, d\}$.

This follows because v can fail to be on the $\mathcal{KDS}$ only if either some point in $\mathcal{V}_a$ or some point in $\mathcal{V}_b$ k-dominates it. If some point in $\mathcal{V}_a$ were to k-dominate v, it has to do so using all k dimensions from the set $\{2, 3, \ldots, d\}$ since v already dominates each point of $\mathcal{V}_a$ on dimension 1. Condition (1) computes the probability that no $u \in \mathcal{V}_a$ k-dominates v. On the other hand, since each point $u \in V_b$ already dominates v on the first coordinate, it needs to only find $k-1$ other dimensions among $\{2, 3, \ldots, d\}$ to achieve k-domination of v. Condition (2) computes the probability that no $u \in \mathcal{V}_b$ k-dominates v. The two events are independent, and so the probability that v belongs to the $\mathcal{KDS}$ is the *product* of these two probabilities. The following two lemmas derive these probabilities.

Lemma 2 *The probability of event (1) is*

$$\frac{A(i, d-1, k)}{i}.$$

Proof. Consider the $i \times (d-1)$ array consisting of the first i rows and the last $(d-1)$ columns of $\mathcal{V}$. This is a random set of i points in $(d-1)$-dimensional space, which for convenience we call the *reduced space*. By induction, the expected $\mathcal{KDS}$ size for this set is $A(i, d-1, k)$. The probability that v is one of these skyline points is $A(i, d-1, k)/i$, by symmetry. Since v already dominates all the points of $\mathcal{V}_a$ in the first coordinate, it is on the $\mathcal{KDS}$ of $\mathcal{V}_a \cup \{v\}$ with the same probability. □

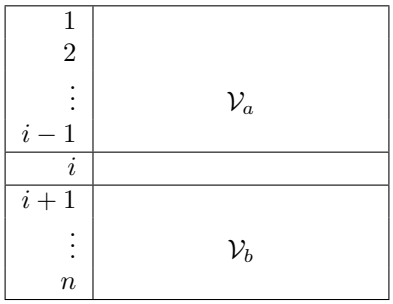


Figure 1: Average case analysis of $\mathcal{KDS}$.

Lemma 3 *The probability of event (2) is*

$$\frac{A(n-i+1, d-1, k-1)}{n-i+1}.$$

Proof. The proof is similar to Lemma 1. □

Therefore, the probability that $v \in \mathcal{KDS}$ is

$$\frac{A(i, d-1, k)}{i} \times \frac{A(n-i+1, d-1, k-1)}{n-i+1}.$$

By summing over all n points, we get

$$A(n, d, k) = \sum_{i=1}^{n} \left(\frac{A(i, d-1, k)}{i} \times \frac{A(n-i+1, d-1, k-1)}{n-i+1} \right)$$

Since $\frac{A(i,d-1,k)}{i}$ is a probability in this recurrence, we can replace it with 1 and derive the following upper bound:

$$A(n, d, k) \leq \sum_{i=1}^{n} \left(\frac{A(n-i+1, d-1, k-1)}{n-i+1} \right),$$

which by a change of index can be rewritten as:

$$A(n, d, k) \leq \sum_{i=1}^{n} \frac{A(i, d-1, k-1)}{i}$$

The function $A(n, d, k)$ is monotone non-decreasing with n because

$$A(n, d, k) = A(n-1, d, k) + \frac{A(n, d-1, k)}{n} \times \frac{A(1, d-1, k-1)}{1}$$

where the second term is non-negative. Therefore, we have the following upper bound:

$$\begin{aligned} A(n, d, k) &\leq \sum_{i=1}^{n} \frac{A(i, d-1, k-1)}{i} \\ &\leq A(n, d-1, k-1) \times \sum_{i=1}^{n} \frac{1}{i} \\ &\leq A(n, d-1, k-1) \times H_n, \end{aligned}$$

where $H_n \approx \ln n$ is the harmonic number [8]. After j iterations, we get:

$$A(n, d, k) \leq A(n, d-j, k-j) \times (H_n)^j$$

The stopping condition for the recurrence is reached when $2(k-j)$ becomes less than or equal to $(d-j)+1$, at which point one can show that the skyline size drops to at most 1. We can show that the maximum number of iterations j is $j = 2k-(d+1)$, which leads to the following theorem.

Theorem 4 *Let $\mathcal{V}$ be a set of n random points in d-dimensional space under the attribute distinctness and statistical independence model. Then, the expected cardinality of their k-dominant skyline is*

$$\begin{cases} O\left((\log n)^{2k-d-1}\right) & \text{for } k > \frac{d+1}{2} \\ O(1) & \text{otherwise} \end{cases}$$

3 Computing the K-Dominant Skyline

Unlike full-dimensional dominance, the k-dominance relation is *not transitive*: that is, $a \succ_k b$ and $b \succ_k c$ does not guarantee $a \succ_k c$. The failure of transitivity means that the algorithms for traditional skylines [1, 10] cannot be used for computing $\mathcal{KDS}$. In fact, to the best of our knowledge, no subquadratic time algorithm is known for k-dominant skylines even for a constant dimension.

Let $I_k \subset \{1, 2, \ldots, d\}$ be an index set of size k, namely, $|I_k| = k$. There are $\binom{d}{k}$ size k distinct

index sets, and each such set defines a subset of k dimensions. The projection of the input set of points $\mathcal{V}$ along any particular set I_k is called a *k-projection* of $\mathcal{V}$. A k-projection is a set of k-dimensional points, and we refer to its skyline as the skyline of the k-projection. Then, the following simple observation is the key to our algorithm.

Lemma 5 *A point $v \in \mathcal{V}$ belongs to $\mathcal{KDS}(\mathcal{V}, k)$ if and only if v belongs to the skylines of all distinct k-projections of $\mathcal{V}$.*

We can, therefore, compute $\mathcal{KDS}(\mathcal{V}, k)$ by computing the skylines of all distinct k-projections of $\mathcal{V}$ and taking their common intersection. In particular, we have the following theorem.

Theorem 6 *Let $\mathcal{V}$ be a set of n points in d dimensions, where $d = O(\log n / \log\log n)$. The k-dominant skyline $\mathcal{KDS}(\mathcal{V}, k)$ is precisely the common intersection of the skylines of all possible k-projections of $\mathcal{V}$, and it can be computed in worst-case time $O(n \log^{d-1} n)$.*

Proof. The number of distinct k-projections of $\mathcal{V}$ is $\binom{d}{k} = \binom{d}{d-k}$. We can compute the skyline of each of these projections in time $O(n \log^{k-1} n)$ using the divide-and-conquer algorithm of [1]. The size of each of these skylines is at most n, and therefore we can compute their common intersection in $O(nd)$ time. The total running time of the algorithm is therefore $O\left(\binom{d}{d-k}\, n \log^{k-1} n\right)$. Since $\binom{d}{d-k} \leq (\frac{ed}{d-k})^{d-k} = O(d^{d-k})$, we can upper bound the running time as $O\left(d^{d-k} n \log^{k-1} n\right)$, which is sub-quadratic as long as $d \leq c \log n / \log\log n$, for an appropriate constant c. □

References

[1] J. L. Bentley. Multidimensional Divide and Conquer. *CACM*, 23(4):214–229, 1980.

[2] J. L. Bentley, H. T. Kung, M. Schkolnick, and C. D. Thompson. On the Average Number of Maxima in a Set of Vectors and Applications. *Journal of the ACM*, 25(4):536–543, 1978.

[3] S. Börzsönyi, D. Kossmann, and K. Stocker. The skyline operator. In *Proc. ICDE*, pages 421–430, 2001.

[4] C. Buchta. On the average number of maxima in a set of vectors. *Information Processing Letters*, 33(2):63 – 65, 1989.

[5] C. Y. Chan, H. V. Jagadish, K. L. Tan, A. K. H. Tung, and Z. Zhang. Finding K-dominant Skylines in High Dimensional Space. In *Proc. ACM SIGMOD*, pages 503–514, 2006.

[6] C. Y. Chan, H. V. Jagadish, K. L. Tan, A. K. H. Tung, and Z. Zhang. On High Dimensional Skylines. In *Proc. EDBT*, pages 478–495, 2006.

[7] S. Chester, A. Thomo, S. Venkatesh, and S. Whitesides. Computing k-regret Minimizing Sets. *PVLDB*, 7(5):389–400, 2014.

[8] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, 2nd edition*. MIT Press, McGraw-Hill Book Company, 2000.

[9] P. Godfrey. Skyline cardinality for relational processing. In *Proc. Foundations of Information and Knowledge Systems (FoIKS)*, pages 78–97, 2004.

[10] P. Godfrey, R. Shipley, and J. Gryz. Algorithms and analyses for maximal vector computation. *VLDB J.*, 16(1):5–28, 2007.

[11] H. K. Hwang, T. H. Tsai, and W. M. Chen. Threshold phenomena in k-dominant skylines of random samples. *SIAM Journal on Computing*, 42(2):405–441, 2013.

[12] D. Kossmann, F. Ramsak, and S. Rost. Shooting stars in the sky: An online algorithm for skyline queries. In *Proc. VLDB*, pages 275–286, 2002.

[13] H. T. Kung, F. Luccio, and F. P. Preparata. On Finding the Maxima of a Set of Vectors. *Journal of the ACM*, 22(4):469–476, 1975.

[14] Y. Lu, J. Zhao, L. Chen, B. Cui, and D. Yang. Effective skyline cardinality estimation on data streams. In *19th International Conference on Database and Expert Systems Applications*, pages 241–254, 2008.

[15] J. Matousek. Computing dominances in E^n. *Information Processing Letters*, 38(5):277–278, 1991.

[16] E. Tiakas, A. N. Papadopoulos, and Y. Manolopoulos. On estimating the maximum domination value and the skyline cardinality of multi-dimensional data sets. *Int. J. Knowledge-Based Organ.*, 3(4):61–83, 2013.

Largest and Smallest Area Triangles on a Given Set of Imprecise Points

Vahideh Keikha* Maarten Löffler† Ali Mohades *

Abstract

In this paper we study the following problem: we are given a set of imprecise points modeled as parallel line segments, and we wish to place three points in different regions such that the resulting triangle has the largest or smallest possible area. We first present some facts about this problem, then we show that for a given set of line segments of equal length the largest possible area triangle can be found in $O(n \log n)$ time, and for line segments of arbitrary length the problem can be solved in $O(n^2)$ time. We also show that the smallest possible area triangle for a set of arbitrary length line segments can be found in $O(n^2)$ time.

1 Introduction

In this paper we study a traditional problem in computational geometry in an imprecise context. Let P be a set of points. We wish to find a sequence of k points in P such that if we connect them into a polygon Q, Q has specific attributes, e.g., Q has the largest or smallest possible area or perimeter, or Q is an empty k-gon, etc. Of course for a given set P such a k-gon does not necessarily exist, for $k > 3$.

1.1 Related Work

Numerous papers studied such problems previously. Dobkin and Snyder [5] presented a linear time algorithm for finding the largest area triangle inscribed in a convex polygon.

Boyce *et al.* [3] presented a dynamic programming algorithm for finding the largest possible area and perimeter convex k-gon on a given set of n points in $O(kn \log n + n \log^2 n)$ time. Aggarwal *et al.* [2] improved their result to $O(kn + n \log n)$.

Due to more applicability, there are more studies concerned with the problem of finding the minimum possible area and perimeter k-gon. Dobkin *et al.* [4] presented an $O(k^2 n \log n + k^5 n)$ time algorithm for finding minimum perimeter k-gons. Their algorithm was improved upon by Aggarwal *et al.* to $O(n \log n + k^4 n)$ time [1]. Eppstein *et al.* [7] studied three problems: finding the smallest possible k-gon, finding the smallest empty k-gon, and finding the smallest possible convex polygon on exactly k points, where the smallest means the smallest possible area or perimeter. They presented a dynamic programming approach with $O(kn^3)$ time and $O(kn^2)$ space, that can also solve the maximization version of the problem as well as some other related problems. Afterwards, Eppstein [8] presented an algorithm for minimum area k-gon problem that runs in $O(n^2 \log n)$ time and $O(n \log n)$ space for constant values of k.

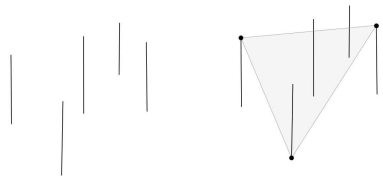


Figure 1: The largest possible area triangle for a set of line segments.

1.2 Problem Definition

We are given a set $L = \{L_1, L_2, \ldots, L_n\}$ of imprecise points modeled as parallel line segments, that is, every segment L_i contains exactly one point $P_i \in L_i$. This gives a point set $P = \{P_1, P_2, \ldots, P_n\}$, and we want to find the largest area or smallest area triangle in P, T_{max} and T_{min}. But because L is a set of imprecise points, we do not know where P is and what could be the possible values of the area. But there should be a lower bound and an upper bound and we are interested in computing these values. So, in this paper we compute the largest possible area of T_{max} and the smallest possible area of T_{min}. We named these problems *MaxMaxArea* and *MinMinArea*, respectively. An illustration of these problems can be seen in Figure 1. In this example, the solution of MinMinArea is zero, as we can find three collinear points.

1.3 Results

We show that

- MaxMaxArea can be solved in $O(n \log n)$ time and $O(n^2)$ time, respectively, for a given set of equal length and arbitrary length parallel line segments, and

*Laboratory of Algorithms and Computational Geometry, Department of Mathematics and Computer Science, Amirkabir University of Technology, Tehran, Iran, va.keikha@aut.ac.ir, mohades@aut.ac.ir

†Department of Information and Computing Sciences, Utrecht University, Utrecht, The Netherlands, m.loffler@uu.nl

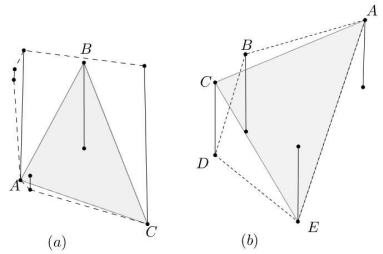


Figure 2: (a) The maximum area true triangle selects its vertices from the endpoints of regions, but not necessarily on the convex hull. (b) Maximum area convex hull does not contain maximum area triangle.

- MinMinArea can be solved in $O(n^2)$ time.

2 Preliminaries

In this section we first present related previous results that may be applicable to our problem, and then discuss some difficulties that occur when dealing with imprecise points.

Boyce *et al.* [3] defined a *rooted* polygon as a polygon with one of its vertices fixed at a given point. In the context of imprecise points, we define the *root* as a given point in a specific region. In this case we throw out the remainder of the root's region and try to find the other vertices of the k-gon in the other $n-1$ remaining regions, and a *rooted* polygon will be a polygon with one of its vertices fixed at a given point in a specific region. Boyce *et al.* [5] showed that the rooted largest area triangle can be found in linear time. They showed that the largest area k-gon only uses points on the convex hull (if there exist at least k points on the convex hull); also Löffler and van Kreveld [9] proved that the maximum area convex polygon always selects its vertices from the endpoints of the line segments, so one may think that the maximum area triangle selects its vertices from the endpoints of regions on the convex hull. But that it is not the case, as can be seen in Figure 2(a).

Also unlike in the precise context, the largest area triangle is not inscribed in the largest possible convex hull of the given set of imprecise points, as illustrated in Figure 2(b).

This problem is more complicated for larger values of k, as illustrated in Figure 3(a); even for k=4, we cannot find the area of the largest strictly convex k-gon, as the angle at a approaches π and we can enlarge the area of the convex 4-gon arbitrarily.

3 Maximum Area Triangle

In the following we first define some notation that we will use in subsequent sections. Let Z be the set of all endpoints of L, $Z = \{L_1^-, L_1^+, L_2^-, L_2^+, \ldots, L_n^-, L_n^+\}$, where L_i^+ denotes the upper endpoint of L_i, and L_i^- denotes the lower endpoint of L_i. We define $C_0 = CH(Z)$ as the convex hull of Z, and $C_1 = CH(Z \setminus C_0)$. By *true triangle* we mean a triangle constructed on three different regions. We assume the segments to be oriented vertically.

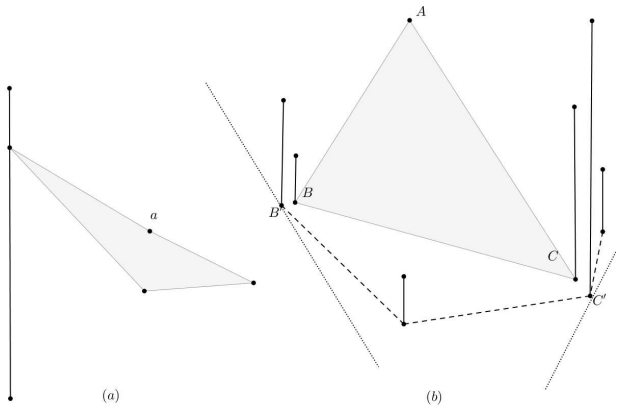


Figure 3: (a)The maximum area convex 4-gon constructed on a set consists of one imprecise and three fixed points, where the inner angle at a approaches π. (b)The largest area true triangle selects at least two vertices from the vertices of C_0.

Observation 1 *If at most two separate regions appear on C_0, there is an optimal solution to the MaxMaxArea problem, such that all the vertices are chosen at endpoints of the line segments, and the two regions which appear on C_0, always appear on the largest possible area true triangle.*

In this case, the largest possible area true triangle can be found in $O(n)$ time. From now on assume more than two different regions appear on C_0.

3.1 Stable Triangle

Dobkin and Snyder [5] defined a *stable* triangle ABC as a triangle with the root A fixed, such that forward advancement of either B or C along the convex hull results in a smaller area, and proved the *Pentagon* lemma (see full version), that is needed for the correctness proof of their algorithm. The area maximizing triangle will be chosen from one of these stable triangles. The idea of their algorithm is to start searching from three consecutive vertices A, B and C on the convex hull. They move C forward until the movement reduces the area, and then move B forward, and again move C, etc. If moving either of them would reduce the area, the triangle is stable and, they move A forward. When A returns to its starting position, they stop the algorithm and report the largest area stable triangle they found. This algorithm has linear running time (see full version for details).

In our imprecise context we define a *stable* triangle in the same way, as a triangle such that forward advancement of either B or C results in a smaller area triangle, but it can be a non-true triangle. So we should be careful that not all the stable triangles are non-true triangles, because then we do not find the largest true triangle among them.

In the following we first show that if we have a convex polygon with vertices from repeated regions (each region appears on the convex hull at most two times), we still can find the solution of MaxMaxArea problem in linear time, then we use this result for designing our algorithms.

4 Largest Area True Triangle

Let ABC be the initial triangle during the execution of the algorithm. Without loss of generality we may assume ABC is a true triangle (as we assume more than two different regions appear on C_0). So, the area of ABC is the initial value of T_{max}. We continue the algorithm naturally, but our movements may result in repeated regions. In full version we show what we do when we encounter repeated regions, and analyse all possible cases that cause a stable non-true triangle. Also, we disscuss the stable true triangle that we accept in each case.

Assume we are given a convex polygon $S = \{s_1, s_2, \ldots, s_n\}$. Similar to the notation of [5], we denote by $\alpha(s_a s_b s_c)$ the stable true triangle which we found during a step of the algorithm, where we started searching from $s_a s_b s_c$. For an arbitrary point $A = s_a$, define the A-rooted maximum true triangle to be $\alpha(s_a s_{a+1} s_{a+2})$. The following lemma states that for the $A+1$-rooted maximum true triangle, it is unnecessary to begin with the collapsed triangle $s_{a+1}s_{a+2}s_{a+3}$. of the A-rooted maximum.

Lemma 1 *If* $\alpha(s_a s_{a+1} s_{a+2}) = (s_a s_b s_c)$, *then* $\alpha(s_{a+1} s_{a+2} s_{a+3}) = \alpha(s_{a+1} s_b s_c)$.

Lemma 2 *Let $S = \{s_1, s_2, \ldots, s_n\}$ be a convex polygon, with vertices from repeated regions. There exist an i ($1 \leq i \leq n$) such that an area maximizing true triangle on s_i is the area maximizing true triangle inscribed in S.*

Corollary 3 *Let L be a set of imprecise points modeled as a set of parallel line segments with arbitrary length. The largest possible area true triangle which selects its vertices from the vertices of C_0 can be found in $O(n \log n)$ time.*

4.1 Equal Length Parallel Line Segments

From the previous section we understand that if we prove that all the candidates points of the vertices of the largest possible area true triangle appear on C_0, and we know that all the possible stable triangles are true triangles, we can directly apply the existing algorithm [5].

Lemma 4 *Let L be a set of equal length parallel line segments. The largest possible area true triangle selects its vertices from the vertices on C_0.*

In case of equal length parallel line segments, when all the upper (and lower) endpoints of the line segments are collinear together, the maximum possible area triangle can be a non-true triangle. In this situation the largest possible area triangle would be constructed on the leftmost and rightmost line segments, and the largest possible area true triangle can be found in linear time. We can determine this situation in $O(n)$ time.

Lemma 5 *Let L be a set of equal length parallel line segments, that is, all the lower (or upper) endpoints are not collinear. The largest pssible area triangle is always a true triangle.*

Theorem 6 *Let L be a set of imprecise points modeled as a set of parallel line segments with equal length. The solution of the problem MaxMaxArea can be found in $O(n \log n)$ time.*

4.2 Arbitrary Length Parallel Line Segments

For simplicity we assume general position, that is, no two vertical line segments have the same x-coordinates. As we saw above, the largest possible area true triangle computed on a set of imprecise points modeled as arbitrary length parallel line segments does not necessarily select its vertices on the convex hull of the regions.

Lemma 7 *Let L be a set of imprecise points modeled as a set of parallel line segments with arbitrary length. At least two vertices of the largest area true triangle are located on C_0 and at most one of its vertices is located on C_1.*

4.2.1 Algorithm

Now we know the combinatorial structure of the largest possible area true triangle: it can select all of its vertices on C_0, or it selects two neighbor vertices on C_0 and one vertex on C_1, or it selects two non-neighbor vertices on C_0 and one vertex on C_1. The largest possible area true triangle is the largest area true triangle among them. In the first case, the largest area true triangle can be found in $O(n \log n)$ time using Corollary 3. In the second case, we try all the edges of C_0 as the base of the triangle. The third vertex can be found by doing a binary search on the

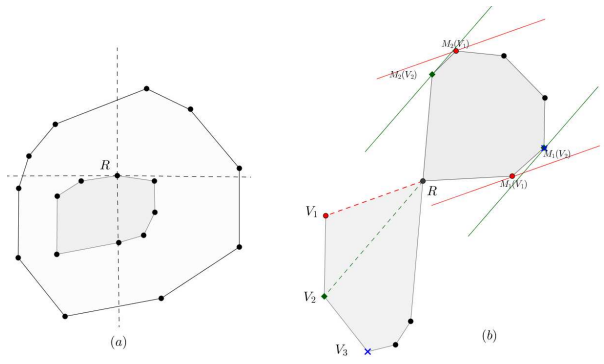


Figure 4: (a) Selection of a point on C_1 as the root R. (b) For the case of diagonal quadrants, for a given $R \in C_1$ and a fixed point V_i in quadrant three, we only need to look for the candidates of the largest area true triangle in one direction, and from $M_1(V_{i-1})$ and $M_2(V_{i-2})$ on C_0.

boundary of C_1. So in this case again we can find the maximum area true triangle in $O(n \log n)$ time. In the third case, assume each of the points of C_1 to be the origin point, R. For every point $R \in C_1$ as the origin, we partition C_0 into four quadrant convex chains, so that the largest area true triangle should be constructed on R and two points on the other quadrants (or only one quadrant), as illustrated in Figure 4(a). If one or two consecutive quadrants include the other vertices of the largest area true triangle, we can find the largest area true triangle in $O(n \log n)$ time by using Corollary 3. Suppose two other vertices are located on diagonal quadrants. Let the cyclic ordering of C_0 be counterclockwise, and let V_1 be the first vertex of quadrant three in the cyclic ordering of C_0. We first find the two candidates points in quadrant one for constructing the largest possible area true triangle on R and V_1, $M_1(V_1)$ and $M_2(V_1)$. For finding $M_1(V_2)$ and $M_2(V_2)$, we just need to start looking from $M_1(V_1)$ and $M_2(V_1)$, etc (see Figure 4(b)). In this case, we can find the largest possible area true triangle in $O(n^2)$ time (see full version for more details).

Theorem 8 *Let L be a set of imprecise points modeled as a set of parallel line segments with arbitrary length. The solution of the problem MaxMaxArea can be found in $O(n^2)$ time.*

5 Smallest Area True Triangle

In this problem, if we find three collinear points on three different input regions, the smallest area triangle would have zero area. We can understand this situation in $O(n^2)$ time. And we cannot hope to do it faster as the problem is 3SUM-hard. In the following we assume that MinMinArea has a non-zero solution.

Lemma 9 *Let L be a set of imprecise points modeled as a set of parallel line segments. Suppose there is no zero-area triangle in L. The smallest area true triangle selects its vertices on the endpoints of the line segments.*

We use the idea of the algorithm presented in [6] that solves the problem in dual space and on an arrangement of lines. As their duality preserves the vertical distances and is order preserving, the minimum area triangle on each vertex in dual space, can be constructed on a line that is located exactly above or below the vertex. In our problem, we need to continue looking in at most two neighbouring faces, when we encounter to repeated regions (see full version for details).

Theorem 10 *Let L be a set of imprecise points modeled as parallel line segments. The solution of the problem MinMinArea can be found in $O(n^2)$ time.*

Acknowledgements

This work was partially supported by the Netherlands Organisation for Scientific Research (NWO) under project no. 614.001.504.

References

[1] A. Aggarwal, H. Imai, N. Katoh and S. Suri. Finding k points with minimum diameter and related problems. *Journal of Algorithms*, 12: 38–56, 1991.

[2] A. Aggarwal, M. M. Klawe, S. Moran, P. Shor and R. Wilber. Geometric applications of a matrix-searching algorithm. *Algorithmica*, 2: 195–208, 1987.

[3] J. E. Boyce, D. P. Dobkin, R. L. Drysdale and L J. Guibas. Finding extremal polygons. *SIAM Journal on Computing*,14 (1), 134–147, 1985.

[4] D.P. Dobkin, R.L. Drysdale, and L.J. Guibas. Finding smallest polygons. *Computational Geometry, Advances in Computing Research*, 1: 181–214, 1983.

[5] D.P. Dobkin and L. Snyder. On a general method for maximizing and minimizing among certain geometric problems. *20th IEEE Symposium on Foundations of Computer Science (FOCS 1979)*, 9–17, 1979.

[6] H. Edelsbrunner, J. O'Rourke and R. Seidel. Constructing Arrangements of Lines and Hyperplanes with Applications. *SIAM Journal on Computing*, 15: 341–363, 1986.

[7] D. Eppstein, M. Overmars, G. Rote and G.J. Woeginger. Finding minimum area k-gons. *Discrete & Computational Geometry*, 7: 45–58, 1992.

[8] D. Eppstein. New algorithms for minimum area k-gons. *3th ACM-SIAM Symposium on Discrete Algorithms (SODA 1992)*, 83–88, 1992.

[9] M. Löffler and M. van Kreveld. Largest and Smallest Convex Hulls for Imprecise Points. *Algorithmica*, 56: 235–269, 2010.

Compact 1-Bend RAC Drawings of 1-Planar Graphs *

Franz J. Brandenburg
University of Passau, 94032 Passau, Germany
brandenb@informatik.uni-passau.de

Abstract

A graph is 1-planar if it can be drawn in the plane so that each edge is crossed at most once. We establish a linear time algorithm that constructs a 1-bend right angle crossing (RAC) drawing on $O(n^2)$ area from a given 1-planar embedding. This improves upon a recent result by Didimo et al. [9] whose drawing algorithm most likely needs high precision arithmetic and exponential area.

1 Introduction

There has been recent interest in beyond-planar graphs, which are classes of graphs that extend the planar graphs and are defined by restrictions on crossings. It is motivated by a correlation between the number and the type of edge crossings and the readability of a graph drawing. Multiple edge crossings and edges that cross at a small angle are bad for a visual analysis of a drawn graph. 1-planar graphs and right angle crossing (RAC) graphs avoid such bad cases. A graph is *1-planar* if it admits a drawing in the plane with at most one crossing per edge. It has been proved multiple times that 1-planar graphs have at most $4n-8$ edges and that the recognition problem is NP-hard, see, e.g., [3] and [10].

Thomassen [14] showed that a 1-planar embedding of a graph cannot be drawn straight line if it includes a B- or a W-configuration, which are displayed in Fig. 1. Then edges cross in an X-configuration. There is a linear time algorithm by Alam et al. [1] that constructs a straight-line drawing of an embedded 3-connected 1-planar graph on a grid of quadratic size with the exception of a single edge in the outer face which may need one bend.

A *right angle crossing* drawing (RAC) is a straight-line drawing so that two edges may cross at a right angle. RAC graphs were introduced by Didimo et al. [8] who showed that such graphs have at most $4n-10$ edges and are incomparable with 1-planar graphs. However, every 1-planar graph admits a *1-bend RAC drawing* [9] so that each edge is represented by at most two segments and segments may cross at a right angle. Graphs with 1-bend RAC drawings have at most $6.5n-13$ edges [2] whereas every graph admits a 3-bend RAC drawing [8]. For their result, Didimo et al. [9] use the convex drawing algorithm of Chiba et al. [4] as a subroutine for drawing planar graphs. That algorithm shows the existence of a convex drawing in a prescribed convex polygon and is supposed to need high precision arithmetic. In addition, Didimo et al. squeeze the drawing of components at a separation pair into a trapezoid whose height is that of a thick line.

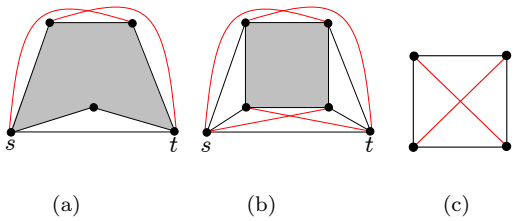


Figure 1: B-, W- and X-configuration

In this work we improve upon the result by Didimo et al. [9] and show that 1-planar graphs admit a 1-bend RAC drawing on $O(n^2)$ area. Our algorithm operates on embeddings of graphs, uses the canonical ordering [6] with an extension to 2-connected graphs [11] and applies the algorithm of Harel and Sardas [11] and Kant [13] for straight-line grid drawings of planar graphs. Pairs of crossing edges are inserted into a planar drawing such that some crossing points are placed on a Thales circle.

2 1-bend RAC Drawings of 1-planar Graphs

In their seminal paper on straight-line drawings of planar graphs, de Fraysseix et al. [6] introduced the canonical ordering of a triangulated planar graph and the shift technique for a straight-line grid drawing. Kant [13] extended the algorithm from triangulated to 3-connected planar graphs both for the computation of the canonical ordering and the placement step so that faces are convex polygons. Harel and Sardas [11] considered 2-connected graphs and implicitly add planar edges towards 3-connectivity. A *canonical ordering* of a planar graph is a numbering of the vertices $(v_1, \ldots, v_n)$ such that $C_n = (v_1, v_n, v_2)$ is the

*Supported by the Deutsche Forschungsgemeinschaft (DFG), grant Br835/18-1

outer face in clockwise order. For $k = 2, \ldots, n-1$, subgraph G_k induced by $v_1, \ldots, v_k$ is 2-connected and the boundary of its exterior face is a cycle C_k including $\{v_2, v_1\}$, called the *contour* of G_k. Vertex v_{k+1} has at least two neighbors in G_k and all neighbors of v_{k+1} in G_k are consecutive in C_k.

A drawing of a simple undirected graph G by the *shift technique* is constructed incrementally by adding the vertices one at a time in the order of the canonical ordering. There is a *contour* from v_1 to v_2 with vertices ordered left to right and above a horizontal line for the edge $\{v_1, v_2\}$. The edges on the contour have slope ± 1. In the k-th step with $k = 3, \ldots, n$, the rightmost neighbor $w_l(v_k)$ of v_k on the contour C_{k-1} is shifted two to the right and v_k is placed at the intersection of the $+1$ diagonal through the leftmost neighbor $w_l(v_k)$ of v_k in C_{k-1} and the -1 diagonal through $w_r(v_k)$. The vertices between $w_l(v_k)$ and $w_r(v_k)$ are shifted one to the right. The shifts can be manipulated efficiently [5, 13].

The main problems are W-configurations and non 3-connected graphs. We address them by a transformation of a given 1-planar embedding into an embedding in normal form.

A *separation pair* $[s, t]$ decomposes a 2-connected graph G into connected components such that $G - \{s, t\} = G_0, \ldots, G_r$ for some $r \geq 1$ and G_i and G_j are not connected for $i \neq j$. We distinguish G_0 as root component by choosing some vertex in G_0. The other components are called *inner components*. By decomposing components recursively, we obtain a decomposition tree T [12] that is a simplification of an SPQR-tree [7].

A 1-planar embedding $\mathcal{E}(G)$ of a graph G is given by the embedding of the planarization $\mathcal{E}(G^\times)$, which is obtained by taking each crossing point as a special vertex of degree four. A B-configuration is the embedding of a component H at a separation pair $[s, t]$ so that the single pair of crossing edges incident to s and t crosses in the outer face, see Fig. 1(a). Vertices s and t can also be in the outer face of G. In a W-configuration, as depicted in Fig. 1(b), there are two pairs of crossing edges incident to s and t one of which crosses in the outer face of H.

A 1-planar embedding $\mathcal{E}(G)$ is in *B-free normal form* if there are no B-configurations, there is a parallel edge in the outer face of each W-configuration that is a copy of the edge between the vertices of the separation pair, and all faces are triangles. The boundary of a face consists of edges including parallel edges and edge segments up to a crossing point. In addition, if there is a quadrangle $Q = (a, b, c, d)$ in counter clockwise order, an ordering of the vertices $a < b < c < d$ and separation pairs $[s, t]$ with $s, t \in \{a, b, c, d\}$, then the inner components are in the lower triangle (a, b, d) of Q.

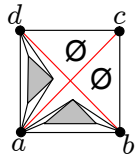


Figure 2: Components inside a quadrangle with crossing edges.

Lemma 1 *There is a linear time algorithm that transforms a 1-planar embedding $\mathcal{E}(G)$ of G into a 1-planar embedding $\mathcal{E}(H)$ of a supergraph H in B-free normal form.*

Proof. The transformation has been established by Alam et al. [1] for 3-connected graphs. First, augment the embedding by as many planar edges as possible. Then remove B-configurations by a flip of the component at the vertices of the separation pair, and again add as many planar edges as possible. The computations are performed on the planarization $\mathcal{E}(G^\times)$, which in the end is triangulated.

For non 3-connected graphs (after the previous steps) consider the decomposition tree induced by separation pairs $[s, t]$. If $G - \{s, t\}$ decomposes into $G_0, \ldots, G_r$ for some $r \geq 1$, then permute the inner components so that two B-configurations $\bar{G}_i$ and $\bar{G}_j$ are placed next to each other and merge them into a W-configuration. Here, $\bar{G}_i = G_i + \{s, t\}$ is the subgraph of G induced by the vertices of G_i and s, t. The remaining W-configurations are separated by parallel edges, which are copies of $\{s, t\}$. A single B-configuration is finally converted into an X-configuration by rerouting $\{s, t\}$. Thereafter, all faces are triangles. Finally, if there is an ordering of the vertices, then inner components are flipped to the right side of the edges of a quadrangle such that they are either in the lower triangle or outside the quadrangle. All steps are performed on the planarization $\mathcal{E}(G^\times)$ and take linear time. □

The *planar skeleton* $\mathcal{E}(G^\square)$ of a 1-planar embedding $\mathcal{E}(G)$ is obtained by removing all pairs of crossing edges. The remaining planar edges keep their embedding. It is something special if the embedding is in B-free normal form. Then $\mathcal{E}(G^\square)$ has triangles and *inner* and *outer quadrangles*. There is an inner quadrangle if it results from an X-configuration of $\mathcal{E}(G)$ and an outer quadrangle if there is a pair of crossing edges between a separation pair and vertices from a component in $\mathcal{E}(G)$. Note that there are vertices in the boundary of an inner quadrangle which result from inner components of separation pairs on the boundary. An outer triangle contains a parallel edge in its boundary. If $[s, t]$ is a separation pair and there is a quadrangle $Q = (s, t, x, y)$ such that

edges $\{s, y\}$ and $\{t, x\}$ cross in $\mathcal{E}(G)$ and the inner components are in Q, then Q is an outer quadrangle and contains the r-th copy of $\{s, t\}$.

Canonical orderings need 3-connected planar graphs [6, 13]. The generalization by Harel and Sardas [11] substitutes missing edges and computes them from the given embedding. In other words, it adds virtual edges. We adopt this technique at separation pairs and the ordering of vertices from inner components. In addition, for every inner quadrangle (a, b, c, d) in counter clockwise order with $a < b < c < d$ in canonical ordering, we keep the edge $\{b, d\}$ as a virtual edge for the placement of d. Canonical orderings can be computed in linear time [11, 13].

We can now state our main result:

Theorem 2 *There is a linear time algorithm that, given 1-planar embedding $\mathcal{E}(G)$, computes a 1-bend RAC drawing of a 1-planar graph G with all vertices on a grid of quadratic size.*

The outline of the algorithm:
Input: A 1-planar embedding $\mathcal{E}(G)$ as a witness of the 1-planarity of a graph G.
Output: A 1-bend RAC drawing of G with vertices on a grid of quadratic size.
1) Compute a 1-planar embedding $\mathcal{E}(H)$ of a supergraph H in B-free normal form from $\mathcal{E}(G)$.
2) Compute a canonical ordering of the planar skeleton $H^{\square}$ using the algorithm by Harel and Sardas [11]. Adjust the embedding so that inner components at separation pairs are flipped into lower triangles.
3) Process the vertices of $H^{\square}$ according to the canonical ordering and draw $H^{\square}$ by the algorithm by Harel and Sardas [11].
4) Insert the crossing edges. If $\{b, d\}$ and $\{a, c\}$ are crossing edges in an inner quadrangle Q in $\mathcal{E}(H)$, then insert them in the quadrangle $Q = (a, b, c, d)$ with a bend and a right angle crossing.

For every separation pair $[s, t]$ with inner components H_i and outer face (a_i, b_i, c_i, d_i) of H_i in the planar skeleton and base $\langle a_i, b_i\rangle$, for $i = 1, \ldots, r$ do steps (a) and (b):

(a) Pick a crossing point q on the Thales circle above (c_i, d_i) in an ϵ-ball around d_i with $\epsilon < 0.5$. Extend the segments (c_i, q) and (d_i, q) beyond q to bend points c_i' and d_i' in the ϵ-ball and connect c_i' and s and d_i' and t by straight segments.

(b) Pick a crossing point p on the Thales circle below (a_i, b_i) in an ϵ-ball around a_i with $\epsilon < 0.5$. Extend the segments (a_i, p) and (b_i, p) beyond p to bend points a_i' and b_i' in the ϵ-ball. Connect b_i' with s and a_i' with t by straight segments.

If there is a quadrangle $Q = (s, t, x, y)$ containing the inner components, then proceed as in (a) at y if

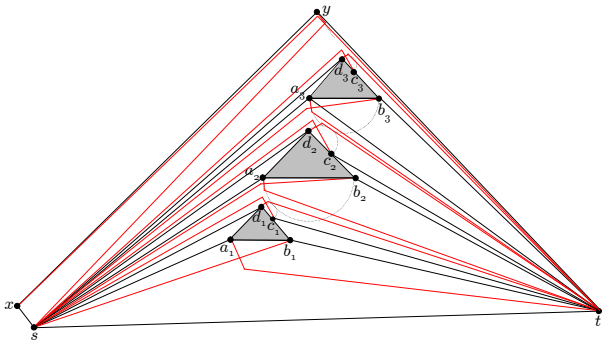


Figure 3: A 1-bend RAC drawing of three inner components with crossing points on a Thales circle

$\langle x\rangle$ and $\langle y\rangle$ with $x < y$ in the canonical ordering and proceed as in (b) at x if there is a chain $\langle x, y\rangle$

Lemma 3 *The correctness of the algorithm is due to the following facts:*

1. *The planar skeleton is drawn planar and straight-line on a grid of size $(2n - 4) \times (n - 2)$.*
2. *Every inner component H is a quadrangle (a, b, c, d) and is drawn as a right triangle with three collinear vertices on a short side and slopes $+1$ and -1 for the short sides.*
3. *Every crossing edge is drawn by at most two segments. A segment is crossed at most once and at a right angle.*

Proof. (Sketch).
The first item is proved in [11]; see also [6, 13].
For the second item, observe that the algorithm preserves the invariant with slopes $-1, +1$ on the contour and an even Manhatten distance between vertices on the contour. For every component H at a separation pair $[s, t]$ the outer boundary of H is a quadrangle (a, b, c, d) and the canonical ordering of H numbers d last. Then vertex d is placed at the intersection of the diagonals through c and a, and c has been placed on the -1 diagonal through b.

Third, if edges $\{a, c\}$ and $\{b, d\}$ cross then there is a quadrangle $Q = (a, b, c, d)$ with $a, b < c < d$ in canonical ordering, then Q may contain drawings of components H with $[a, b]$ and $[a, d]$ as a separation pair. By the canonical ordering, a and b are drawn first, the components H at the separation pair $[a, b]$ are drawn next and subsequently, components H' at $[a, d]$ are added to the drawing. Then the lower right corner w of the drawing of H' is at $(-1, 1)$ from the top corner z of the drawing of H, and both are connected by a straight line with a. Now the first segment of $\{a, c\}$ is routed between w and z. The drawings of H and H' are below the line between d and b, since edge $\{b, d\}$ is used as a virtual edge for the placement

of d. So there is space for a right angle crossing between the second segment of $\{a, c\}$ and a segment of $\{b, d\}$ above the line between b and d.

For every component $H = (a, b, c, d)$ at a separation pair $[s, t]$, the crossing edges $\{s, d\}$ and $\{a, t\}$ are drawn each with a bend in an ϵ-ball at a and with a crossing point on a Thales circle. Similarly, $\{s, c\}$ and $\{b, t\}$ are drawn each with a bend in an ϵ-ball at b. These edges do not cross other edges, since they are routed close to the edges $\{s, a\}$ and $\{s, b\}$ and the lines (a, t) and (b, t). □

The algorithm runs in linear time. Using the data structure from the underlying planarization $\mathcal{E}(G^{\times})$, the normal form embedding of a supergraph H can be computed in linear time. Graph H has parallel edge, but in total H has at most $4n - 8$ edges, since the parallel edge and a pair of edges crossing in the outer face of a component can be substituted by four edges in another 1-planar embedding. The computation of a canonical ordering takes linear time, as does the drawing of the planar skeleton $H^{\square}$ [11,13]. There are at most $n - 2$ pairs of crossing edges, which can each be drawn in $O(1)$ time.

3 Conclusion and Perspectives

Using an extension of the canonical ordering and well-known algorithms for straight-line grid drawings of planar graphs, we have shown that every 1-planar graph admits a 1-bend RAC drawing on a grid of quadratic size. However, the bends and the crossing points are not on grid points. Is this doable on a polynomial size grid?

4 Acknowledgements

I wish to thank the anonymous reviewers for their valuable suggestions and comments.

References

[1] M. J. Alam, F. J. Brandenburg, and S. G. Kobourov. Straight-line drawings of 3-connected 1-planar graphs. In S. Wismath and A. Wolff, editors, *GD 2013*, volume 8242 of *LNCS*, pages 83–94. Springer, 2013.

[2] K. Arikushi, R. Fulek, B. Keszegh, F. Morić, and C. D. Tóth. Graphs that admit right angle crossing drawings. *Comput. Geom.*, 45(4):169–177, 2012.

[3] R. Bodendiek, H. Schumacher, and K. Wagner. Über 1-optimale Graphen. *Mathematische Nachrichten*, 117:323–339, 1984.

[4] N. Chiba, T. Yamanouchi, and T. Nishizeki. Linear time algorithms for convex drawings of planar graphs. In D. Z. Chen and D. T. Lee, editors, *Progress in Graph Theory*, Academic Press, pages 153–173, 1984.

[5] M. Chrobak and T. Payne. A linear-time algorithm for drawing a planar graph on a grid. *Inform. Process. Lett.*, 54:241–246, 1995.

[6] H. de Fraysseix, J. Pach, and R. Pollack. How to draw a planar graph on a grid. *Combinatorica*, 10:41–51, 1990.

[7] G. Di Battista and R. Tamassia. On-line planarity testing. *SIAM J. Comput.*, 25(5):956–997, 1996.

[8] W. Didimo, P. Eades, and G. Liotta. Drawing graphs with right angle crossings. *Theor. Comput. Sci.*, 412(39):5156–5166, 2011.

[9] W. Didimo, G. Liotta, S. Mehrabi, and F. Montecchiani. 1-bend RAC drawings of 1-planar graphs. In Y. Hu and M. Nöllenburg, editors, *GD 2016*, volume 9801 of *Lecture Notes in Computer Science*, pages 335–343. Springer, 2016.

[10] A. Grigoriev and H. L. Bodlaender. Algorithms for graphs embeddable with few crossings per edge. *Algorithmica*, 49(1):1–11, 2007.

[11] D. Harel and M. Sardas. An algorithm for straight-line drawing of planar graphs. 20:119–135, 1998.

[12] J. E. Hopcroft and R. E. Tarjan. Dividing a graph into triconnected components. *SIAM J. Comput.*, 2(3):135–158, 1973.

[13] G. Kant. Drawing planar graphs using the canonical ordering. *Algorithmica*, 16:4–32, 1996.

[14] C. Thomassen. Rectilinear drawings of graphs. *J. Graph Theor.*, 12(3):335–341, 1988.

Non-crossing drawings of multiple geometric Steiner arborescences*

Irina Kostitsyna† Bettina Speckmann‡ Kevin Verbeek†

1 Introduction

An important problem in the area of computational geometry is the Euclidean Steiner Tree problem: given a set of n points in the plane, find a set of line segments that connect all points in a single connected component, such that the total length of the line segments is minimized. The Euclidean Steiner Tree problem and variants thereof have many applications in practice. For example, rectilinear Steiner trees, where line segments must be horizontal or vertical, are commonly used for wire routing in VLSI design. Also of interest are Steiner arborescences [5]: Steiner trees rooted at a node r, such that the path in the Steiner tree between r and any other input point must be a shortest path with respect to some metric (see Fig. 1). A variant of Steiner arborescences, namely angle-restricted Steiner arborescences, or *flux trees*, have recently been used to design flow maps [4]. In a flux tree, the path from an input point to the root r must always go roughly in the direction of r, that is, it can only deviate by at most a fixed angle.

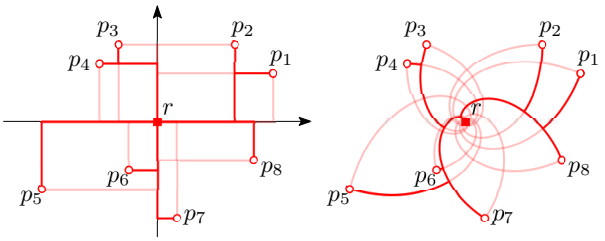


Figure 1: A rectilinear Steiner arborescence and a flux tree with eight terminals.

The Euclidean Steiner tree problem and its variants have been studied extensively. Although most of these problems are NP-hard, many efficient approximation algorithms are known [2, 8]. However, if we want to compute multiple Steiner trees for multiple point sets, such that the Steiner trees have no or few crossings, then there are very few results. Aichholzer *et al.* [1] give an algorithm that, given two sets of n points in the plane, computes in $O(n \log n)$ time two spanning trees (not Steiner trees) such that the diameters of the trees and the number of intersections between the trees are small. Similar (weaker) results have also been obtained for drawing more than two plane spanning trees with few crossings [6, 7]. Recently, Bereg *et al.* [3] presented approximation algorithms for computing k disjoint Steiner trees for k point sets, with approximation ratios $O(\sqrt{n} \log k)$ and $k + \varepsilon$ for general k, $(5/3 + \varepsilon)$ for $k = 3$, and a PTAS for $k = 2$.

In this paper we consider multiple Steiner arborescences. Two or more non-crossing Steiner arborescences need not even exist. Nonetheless, they are very relevant in practice, for example for constructing flow maps. A flux tree can only show information about one source, but ideally multiple sources should be shown simultaneously, in such a way that the corresponding flux trees have few or no crossings. To the best of our knowledge, these problems have not been studied.

Problem statement. We study the following problem: given a set of k roots $r_1, \ldots, r_k \in \mathbb{R}^2$, and k sets of terminals $T_1, \ldots, T_k \subset \mathbb{R}^2$, do there exist k non-crossing Steiner arborescences which connect each set of terminals T_i to its root r_i? We focus mostly on the case $k = 2$. When considering only two trees, we refer to the first tree as the *red* tree, with root r_1 and terminals $T_1 = \{p_1, \ldots, p_n\}$, and the second tree as the *blue* tree with root r_2 and terminals $T_2 = \{q_1, \ldots, q_m\}$. We consider both rectilinear Steiner arborescences and flux trees.

Preliminaries. It follows from the definition of geometric Steiner arborescences that the path between the root and a terminal must completely lie in a particular region. For rectilinear Steiner arborescences this is the rectangle spanned by the root and the terminal. For flux trees this region is bounded by two logarithmic spirals and is hence called the *spiral region* [4]. Here we refer to these regions as $\mathcal{R}$*-regions* and denote the $\mathcal{R}$-region between a root r and a terminal t by $\mathcal{R}(r, t)$. When considering multiple rectilinear Steiner arborescences, we allow the sets of axes of the two Steiner arboresences to be different.

We say that two $\mathcal{R}$-regions $\mathcal{R}(r_1, p_i)$ and $\mathcal{R}(r_2, q_j)$ *fully intersect* if $r_1, p_i \notin \mathcal{R}(r_2, q_j)$, $r_2, q_j \notin \mathcal{R}(r_1, p_i)$, and segments $r_1 p_i$ and $r_2 q_j$ intersect. It is easy to verify that two non-crossing Steiner arborescences do not exist if there are two $\mathcal{R}$-regions $\mathcal{R}(r_1, p_i)$ and $\mathcal{R}(r_2, q_j)$ that fully intersect (see Fig. 2): any two paths routed

*I.K. and B.S. are partially supported by the Netherlands Organisation for Scientific Research (NWO) under grant number 639.023.208. K.V. is supported by the Netherlands Organisation for Scientific Research (NWO) under grant number 639.021.541. I.K. is also supported by F.R.S.-FNRS.

†Computer Science Department, Université libre de Bruxelles, Belgium, irina.kostitsyna@ulb.ac.be

‡Dep. of Mathematics and Computer Science, TU Eindhoven, The Netherlands, [b.speckmann|k.a.b.verbeek]@tue.nl

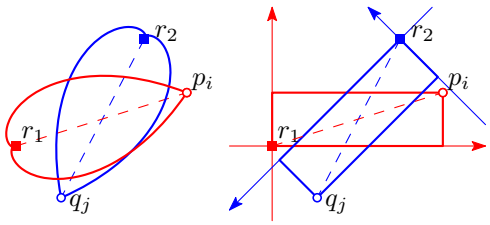


Figure 2: Two $\mathcal{R}$-regions fully intersect.

within the respective $\mathcal{R}$-regions must intersect.

When drawing (the paths of) Steiner arborescences we consider two models:

(a) *Free turns*: Paths can turn anywhere.

(b) *Limited turns*: Paths can only turn at a Steiner point or at a corner of an $\mathcal{R}$-region (as in Fig. 1).

The limited turns model can be quite restrictive. Fig. 3 shows an example where a non-crossing drawing of two rectilinear Steiner arborescences exists only in the free turns model.

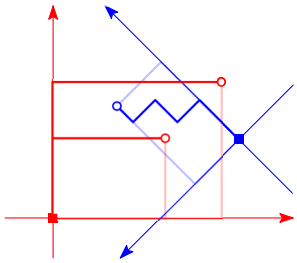


Figure 3: Non-crossing drawing exists only in free turns model.

Results. In the free turns model, we show in Section 2.1 that two rectilinear Steiner arborescences have a non-crossing drawing if (a) no two $\mathcal{R}$-regions fully intersect, and (b) the roots are not contained inside any $\mathcal{R}$-region. In Section 2.2 we lift the constraint on the roots and show how to reduce the decision problem to 2SAT. In Section 3 we show that in the limited turns model it is NP-hard to decide whether multiple rectilinear Steiner arborescences have a non-crossing drawing. The setting of flux trees is more subtle. Our NP-hardness result extends, but testing whether there exists a non-crossing drawing requires additional conditions to be fulfilled (see Section 4). Due to space limitations, some figures and proofs are omitted from this short abstract and can be found in the full version of the paper.

2 Two rectilinear Steiner arborescences

In this section we show how to decide if a non-crossing drawing of two rectilinear Steiner arborescences in the free turns model exists, and how to construct such a drawing. We consider the general case, when the axes of the two arborescences are not aligned. The free

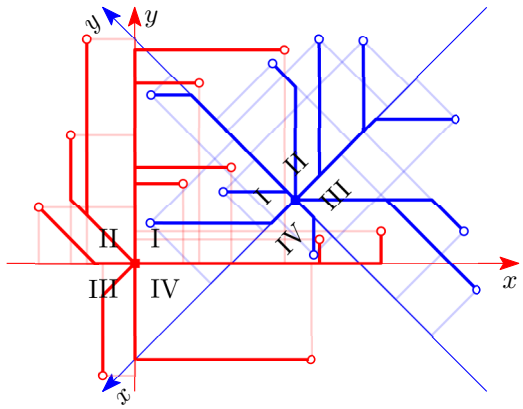


Figure 4: Non-crossing drawing of two rectilinear Steiner arborescences.

turn model implies that, in principle, the paths of the trees can approximate any xy-monotone curve. We show that we can in fact restrict the directions of the paths to the 8 directions implied by the axes of the two rectilinear Steiner arborescences (see Fig. 4).

2.1 Roots not contained in $\mathcal{R}$-regions

Consider the four quadrants of the coordinate system of the red arborescence ordered counter-clockwise, and the four quadrants of the blue arborescence ordered clockwise. Let the first quadrants face each other (see Fig. 4). There are eleven faces in the arrangement of the four coordinate axes, to which we refer by the two corresponding quadrants. For simplicity of presentation, we assume that no terminal lies on an axis of the other color. Let $\mathcal{C}_b$ be a cone in the red coordinate system with the apex in the blue root and with angle range $[0, \frac{\pi}{2}]$, and let $\mathcal{C}_r$ be a cone in the blue coordinate system with the apex in the red root and with angle range $[0, \frac{\pi}{2}]$. If the roots are not contained in the $\mathcal{R}$-regions of the other tree then there are no red terminals in $\mathcal{C}_b$, and there are no blue terminals in $\mathcal{C}_r$.

Given a red terminal p, and some xy-monotone path π_p connecting p to r_1, define a *dead region* $\mathcal{D}_2(\pi_p)$, with respect to the blue root r_2, to be the union of all points q such that path π_p intersects region $\mathcal{R}(r_2, q)$ and disconnects q from r_2. Analogously, define a *dead region* $\mathcal{D}_1(\pi_q)$ for a blue terminal q.

Observe that π_p is on the boundary of $\mathcal{D}_2(\pi_p)$, and that the rest of its boundary consists of lines parallel to blue axes. For example, in Fig. 5, $\mathcal{D}_2(\pi_p)$ is bounded by two lines parallel to the blue y-axis that go through r_1 and p (as p lies in the blue quadrant II). If p were, for example, in quadrant I, than the bounding line passing through p would be parallel to the blue x-axis.

Given a red terminal p such that $\mathcal{R}(r_1, p)$ does not contain r_2, define the *dead region* $\mathcal{D}_2(p)$ to be the intersection of dead regions $\mathcal{D}_2(\pi_p)$ for all possible paths π_p connecting p to r_1, i.e., $\mathcal{D}_2(p) = \bigcap_{\pi_p} \mathcal{D}_2(\pi_p)$. Define

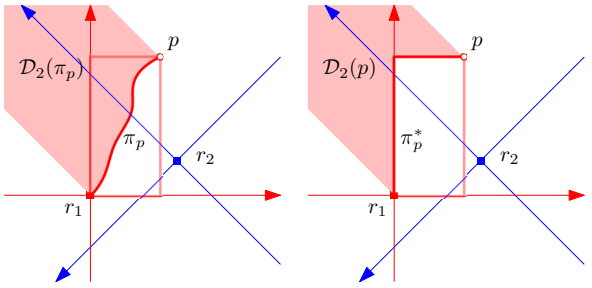


Figure 5: Dead regions of a path π_p and a terminal p.

red terminals in		blue terminals in
(a) I ∩ II	vs.	(b) II ∩ I,
(c) I ∩ III	vs.	(d) III ∩ I,
(e) I ∩ IV	vs.	(f) IV ∩ I,
(g) III ∩ IV	vs.	(h) IV ∩ III,
(i) I ∩ III	vs.	(j) IV ∩ IV,
(k) IV ∩ IV	vs.	(l) III ∩ I.

Table 1: Mutually exclusive cases of locations of red and blue terminals.

dead region $\mathcal{D}_1(q)$ analogously. From this definition it follows that:

Proposition 1 *Let a red terminal $p \notin \mathcal{R}(r_2, q)$ and a blue terminal $q \notin \mathcal{R}(r_1, p)$. Then $q \in \mathcal{D}_2(p)$ if and only if $\mathcal{R}(r_1, p)$ fully intersects $\mathcal{R}(r_2, q)$, and therefore $q \in \mathcal{D}_2(p)$ if and only if $p \in \mathcal{D}_1(q)$.*

There can be terminals whose dead regions are empty. For example, if $p \in$ I∩I, then there is a path connecting p to r_1 that does not obstruct routing of any possible blue terminal. Consider the eight faces of the axes arrangement except for faces I ∩ I, I ∩ IV, and IV ∩ I. For terminals p and q in them, $\mathcal{D}_2(p)$ and $\mathcal{D}_1(q)$ are not empty. Moreover, in these faces $p \in \mathcal{D}_2(p)$ and $q \in \mathcal{D}_1(q)$. Denote π_p^* to be the path that connects p to r_1 along the boundary of $\mathcal{D}_2(p)$, and π_q^* to be the path that connects q to r_2 along the boundary of $\mathcal{D}_1(q)$ (see Fig. 5 (right)). We can show that:

Proposition 2 *Paths π_p^* and π_q^* are xy-monotone in the red and blue coordinate systems, respectively.*

Therefore π_p^* and π_q^* are valid paths connecting p to r_1 and q to r_2. From Proposition 1 it follows that if a blue terminal $q \notin \mathcal{D}_2(p)$ then π_p^* does not itersect π_q^*. In the full version of the paper we carefully go through all cases for terminals p and q such that the corresponding dead regions $\mathcal{D}_2(p)$ and $\mathcal{D}_1(q)$ are not empty and their boundaries contain paths connecting the terminals to their roots.

Routing rules. Notice that two cases, when there is a red terminal p in I ∩ II, and when there is a blue terminal q in II∩I, are mutually exclusive, for otherwise $\mathcal{R}(r_1, p)$ would fully intersect $\mathcal{R}(r_2, q)$. Table 1 gives a full list of all mutually exclusive cases. Given two roots and two sets of terminals such that no two $\mathcal{R}$-regions of opposite colors fully intersect, we can construct two non-intersecting Steiner arborescences using simple routing rules (see Fig. 4). First, red terminals p in (II ∪ III)\$\mathcal{C}_r$, I ∩ II, I ∩ III, IV ∩ III, or IV ∩ IV are routed along π_p^*. Blue terminals q in (II ∪ III)\$\mathcal{C}_b$, II ∩ I, III ∩ I, III ∩ IV, or IV ∩ IV are routed along π_q^*. Next, terminals in $\mathcal{C}_r$ and $\mathcal{C}_b$ are routed as shown in Fig. 6. Lastly, the rest of the terminals are routed so as to avoid already constructed paths. Detailed routing rules can be found in the full version.

Theorem 3 *Two rectilinear Steiner arborescences can be drawn with no crossings in the free turn model if no two $\mathcal{R}$-regions fully intersect and if no roots are contained in $\mathcal{R}$-regions.*

2.2 Roots contained in $\mathcal{R}$-regions

Next, we relax the restriction that the roots cannot be contained in $\mathcal{R}$-regions. Now, for any $\mathcal{R}$-region that contains the root of the other color, we need to make a choice of how to route the terminal-to-root path around the other root. This choice clearly can affect later decisions. Before we proceed, we need some additional definitions.

Points r and t split the boundary of $\mathcal{R}(r, t)$ into two components that we call the *right* side $\sigma^+(r, t)$ (that leaves the $\mathcal{R}$-region to the left if moving from r to t), and the *left* side $\sigma^-(r, t)$.

We say that $\mathcal{R}(r_1, p_i)$ *cuts* the right (left) side of $\mathcal{R}(r_2, q_j)$, if $r_1 \in \mathcal{R}(r_2, q_j)$, and both sides of $\mathcal{R}(r_1, p_i)$ intersect the right (left) side of $\mathcal{R}(r_2, q_j)$ (see Fig. 7). We can now define a dead region for a terminal p for a fixed direction a p-to-r_1 path must take around r_2.

Given a red terminal p, define the *left (right) dead region* $\mathcal{D}_2^l(p)$ ($\mathcal{D}_2^r(p)$) to be the intersection of dead regions $\mathcal{D}_2(\pi_p)$ for all possible paths π_p connecting p to r_1 that go around r_2 from the left (right), i.e., $\mathcal{D}_2^l(p) = \bigcap_{\text{left } \pi_p} \mathcal{D}_2(\pi_p)$ and $\mathcal{D}_2^r(p) = \bigcap_{\text{right } \pi_p} \mathcal{D}_2(\pi_p)$. Analogously, define $\mathcal{D}_1^l(q)$ and $\mathcal{D}_1^r(q)$. Note that in this definition we do not require $\mathcal{R}$-regions to be root free. We can make an observation similar to the one in the

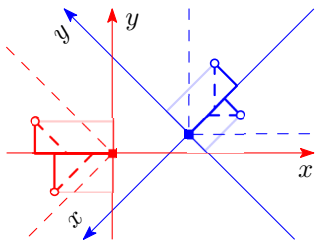


Figure 6: Routing rule for terminals in $\mathcal{C}_r$ and $\mathcal{C}_b$.

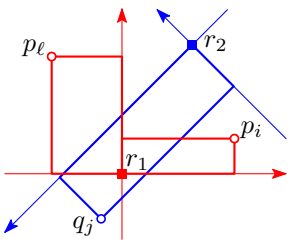


Figure 7: Red $\mathcal{R}$-regions cut the left and right side of the blue $\mathcal{R}$-region.

previous section. Let the blue root $r_2 \in \mathcal{R}(r_1, p)$ and the red root $r_1 \notin \mathcal{R}(r_2, q)$. A blue terminal $q \in \mathcal{D}_2^l(p)$ ($q \in \mathcal{D}_2^r(p)$) if and only if $\mathcal{R}(r_2, q)$ fully intersects the left side $\sigma^+(r_1, p)$ (right side $\sigma^-(r_1, p)$) of $\mathcal{R}(r_1, p)$. Therefore, if $r_2 \in \mathcal{R}(r_1, p)$ and $r_1 \notin \mathcal{R}(r_2, q)$, the blue terminal $q \notin \mathcal{D}_2^l(p)$ ($q \notin \mathcal{D}_2^r(p)$) if and only if $\mathcal{D}_1(q)$ does not intersect $\mathcal{D}_2^l(p)$ ($\mathcal{D}_2^r(p)$). We can extend this observation to the case where the blue root $r_2 \in \mathcal{R}(r_1, p)$ and the red root $r_1 \in \mathcal{R}(r_2, q)$:

Observation 1 *If $r_2 \in \mathcal{R}(r_1, p)$ and $r_1 \in \mathcal{R}(r_2, q)$, the blue terminal $q \notin \mathcal{D}_2^l(p)$ ($q \notin \mathcal{D}_2^r(p)$) if and only if $\mathcal{D}_1^l(q)$ ($\mathcal{D}_1^r(q)$) does not intersect $\mathcal{D}_2^l(p)$ ($\mathcal{D}_2^r(p)$).*

We reduce the problem of choosing the direction of the path with respect to the other root to 2SAT. Given a solution to the 2SAT formula that fixes directions of the paths with terminals in cones $\mathcal{C}_r$ and $\mathcal{C}_b$, we can again route the paths along the boundaries of the dead regions. More details can be found in the full version of the paper.

Theorem 4 *We can decide in polynomial time whether two rectilinear Steiner arborescences can be drawn with no crossings in the free turn model.*

3 Drawing many Steiner arborescences is NP-hard

If the number of arborescences in the problem is not bounded, the problem becomes NP-hard for the limited turns model.

Theorem 5 *It is NP-hard to decide whether k rectilinear Steiner arborescences, where k is part of the input, can be drawn without crossings in the limited turns model, even if all trees are axis-aligned.*

Theorem 6 *It is NP-hard to decide whether k flux trees, where k is part of the input, can be drawn without crossings in the limited turns model.*

4 Two flux trees

In this section we sketch how to draw two flux trees in the free turns model with no root containment in $\mathcal{R}$-regions. Similarly to the rectilinear case, free turns imply that the terminal-to-root paths can approximate any *spiral monotone*[1] curve. Here we restrict the paths to only follow four logarithmic spirals, positive and negative spirals with the origin in the red root, and positive and negative logarithmic spirals with the origin in the blue root. We prove the following theorem in the full version of the paper.

[1] A spiral monotone curve [4] requires that for any point the angle between the tangent and the direction to the destination is not greater than a given parameter α.

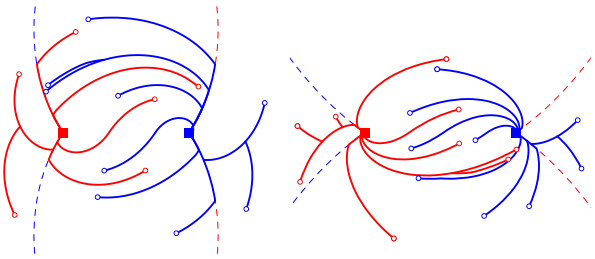


Figure 8: Two non-crossing drawings of flux trees for $\alpha = 60°$ (left) and $\alpha = 30°$ (right).

Theorem 7 *We can decide in polynomial time if two flux trees with no root containment in $\mathcal{R}$-regions can be drawn without crossings in the free turns model.*

Figure 8 shows the final result of the procedure.

References

[1] O. Aichholzer et al. Connecting colored point sets. *Discrete Applied Mathematics*, 155(3), 2007.

[2] S. Arora. Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. *Journal of the ACM*, 45(5), 1998.

[3] S. Bereg et al. Colored non-crossing Euclidean Steiner forest. In *ISAAC*, 2015.

[4] K. Buchin, B. Speckmann, and K. Verbeek. Angle-restricted Steiner arborescences for flow map layout. *Algorithmica*, 72(2), 2015.

[5] J. Cong, A. Kahng, and Kwok-Shing Leung. Efficient algorithms for the minimum shortest path Steiner arborescence problem with applications to VLSI physical design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 17(1), 1998.

[6] M. Kano, C. Merino, and J. Urrutia. On plane spanning trees and cycles of multicolored point sets with few intersections. *Information Processing Letters*, 93(6), 2005.

[7] J. Leaños et al. Spanning trees of multicoloured point sets with few intersections. In *IJCCGGT*, 2005.

[8] B. Lu and L. Ruan. Polynomial time approximation scheme for the rectilinear Steiner arborescence problem. *Journal of Combinatorial Optimization*, 4(3), 2000.

Towards a Topology-Shape-Metrics Framework for Ortho-Radial Drawings

Lukas Barth* Benjamin Niedermann* Ignaz Rutter† Matthias Wolf*

Abstract

Ortho-Radial drawings are a generalization of orthogonal drawings to grids that are formed by concentric circles and straight-line spokes from the center.

We show that bend-free planar ortho-radial drawings can be combinatorially described in terms of the distribution of the angles around the vertices. Previously, such a characterization was only known for paths, cycles, and theta graphs [5], and in the special case of rectangular drawings for cubic graphs [4], where the contour of each face is required to be a rectangle. This is an important ingredient in establishing an ortho-radial analogue of Tamassia's Topology-Shape-Metrics Framework for bend minimization in planar orthogonal drawings.

1 Introduction

Grid drawings of graphs map vertices to grid points, and edges to internally disjoint curves on the grid lines connecting their endpoints. *Orthogonal grids*, where the grid lines are horizontal and vertical lines, are popular and widely used in graph drawing. Their strength lies in their simple structure, their high angular resolution, and the limited number of directions. Graphs admitting orthogonal grid drawings must be *4-planar*, i.e., they must be planar and have maximum degree 4.

It is well known that, a bend-free planar orthogonal drawing Γ of a 4-plane graph G, i.e., a 4-planar graph with a fixed combinatorial embedding, can be combinatorially described by the distribution of the angles around the vertices. For any incidence between a vertex v and a face f that lies to the right of the edges uv, vw, we measure the counterclockwise angle $a \in \{90°, 180°, 270°, 360°\}$ between vu and vw. In this way, we assign an angle to each vertex–face incidence. Consider two edges uv, vw not necessarily bounding a common face and let α be the sum of all the angles that lie locally to the right of uvw. We define the rotation of the path uvw as $\mathrm{rot}(uvw) = 2 - \alpha/90°$, i.e., intuitively left and right turns correspond to rotations of -1 and 1, respectively, whereas going straight corresponds to a rotation of 0. We further generalize this to arbitrary paths $P = v_1, \ldots, v_k$ as $\mathrm{rot}(P) = \sum_{i=2}^{k-1} \mathrm{rot}(v_{i-1}v_iv_{i+1})$ and to cycles $C = v_1, \ldots, v_k, v_1$ as $\mathrm{rot}(C) = \sum_{i=1}^{k} \mathrm{rot}(v_{i-1}v_iv_{i+1})$, where $v_0 = v_k$ and $v_{k+1} = v_1$. For a face f, we define $\mathrm{rot}(f) = \mathrm{rot}(C_f)$, where C_f is the boundary of f directed such that f lies to its right. It is not hard to see that an angle assignment stemming from an orthogonal drawing satisfies the following conditions.

1. The sum of the angles around each vertex is 360°.
2. For each internal face f it is $\mathrm{rot}(f) = 4$ and $\mathrm{rot}(f) = -4$ for the outer face.

An angle assignment satisfying these conditions is called *orthogonal representation*. Tamassia [6] showed that, conversely, for any orthogonal representation there exists a corresponding planar orthogonal drawing with the given angles. It is this characterization, which decouples the shape of an orthogonal drawing (described in the form of an orthogonal representation) from its geometric realization, that has enabled a three-step framework for computing orthogonal planar drawings, the Topology-Shape-Metrics (TSM) framework, that is at the heart of various bend minimization algorithms for orthogonal drawings [6, 1, 2, 3]. Note that bends can be seen as subdivision vertices with a 90° and a 270° angle.

The goal of this work is to provide a similar result and thus to establish the existence of an analogous framework for *ortho-radial drawings*, which are based on *ortho-radial grids* formed by concentric circles and spokes emanating from the circles' center c; see Fig. 1a. In this case, our 4-plane input graph G comes with two designated faces, an *outer face*, which shall form the outer face of the drawing and a *central face* whose interior shall contain c. All other faces are *regular*. A simple cycle in G is *essential* if it contains c in its interior, otherwise it is *non-essential*. Throughout this paper we assume that G contains at least one

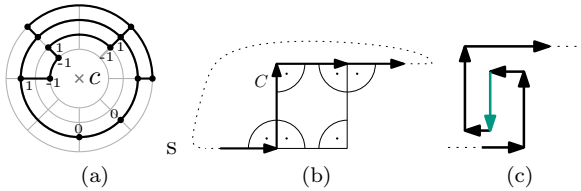


Figure 1: (a) An ortho-radial drawing on an ortho-radial grid, the small numbers give the rotations at the vertices in the central face. (b,c) Ortho-radial representations with locally correct angle sums that cannot be realized; the dotted curve is a single edge.

*Karlsruhe Institute of Technology, Germany, firstname.lastname@kit.edu, Lukas Barth was partially supported by DFG Research Training Group 2153

†TU Eindhoven, The Netherlands, i.rutter@tue.nl

essential cycle. If it does not, then the central and the outer face are identical, and the ortho-radial drawing is equivalent to an orthogonal drawing [5].

It is not hard to see that also ortho-radial drawings induce angle assignments as above, which we call *ortho-radial representations*. It is clear that again the angle sums around each vertex must be 360°, and further similar to the orthogonal case, it is $\mathrm{rot}(f) = 4$ for regular faces and $\mathrm{rot}(f) = 0$ for the central and the outer face (recall that we assume them to be distinct). However, there are examples of such assignments that have no geometric realization; see Fig. 1b. Up to this point a characterization of the ortho-radial representations that have a corresponding drawing has been achieved only for paths, cycles, and theta-graphs [5] and for 3-regular *rectangular* graphs [4], whose ortho-radial representation is such that internal faces have exactly four 90° angles, while all other incident angles are 180°, and the central and outer face have only 180° angles. Our main result is a characterization of the ortho-radial representations of arbitrary 4-plane graphs that correspond to an ortho-radial drawing.

We introduce some notation and our definition of a *valid* ortho-radial representation in Section 2. Afterwards, we first show in Section 3 that valid ortho-radial representations characterize the ortho-radial drawings of rectangular graphs. Based on that special case of 4-planar graphs, we then present the characterization for general 4-planar graphs in Section 4.

2 Preliminaries and Ortho-Radial Representations

In this paper, all paths and cycles are directed. We implicitly direct cycles that do not cross themselves, e.g., facial cycles, such that their interior lies to the right, and we consider a cycle to be part of both its interior and its exterior, i.e., the interior and the exterior are closed. We further assume that paths are *simple* but cycles may be non-simple, though they may contain each edge at most once in each direction.

Making use of the view of ortho-radial drawings as orthogonal drawings of a cylinder, we classify the edges of a drawing as pointing *left*, *right*, *down* or *up*, respectively. Edges pointing left or right are *horizontal* edges and edges pointing up or down are *vertical* edges. Note further, that an ortho-radial representation determines the directions of all edges. Considering again the example from Fig. 1b it can be seen that the essential cycle C contains an up edge but no down edge, and thus there is no corresponding drawing. However, the existence of a down edge on each essential cycle containing an up edge is not sufficient for the existence of a drawing even in the case of cycles; see Fig. 1c. The reason is that, in some sense, the down edge in this case is too wound up to be of any help. Instead we need a somewhat more global measure than up and down, which we introduce next.

Ortho-Radial Representations For a 4-plane graph G with a given ortho-radial representation Γ, we fix an arbitrary reference edge $e^\star = rs$ on the outer face that points to the right, i.e., the outer face lies on its left. Let C be a simple essential cycle and let P be a path from s to a vertex v of C. We now define a labeling of the edges of C with respect to P and $e^\star$ as $\ell_C^P(e) = \mathrm{rot}(e^\star + P + C[v, e])$, where $C[v, e]$ denotes the part of C from v to e. In the following we are mostly interested in labelings with respect to so-called *elementary paths* P, where v is the first vertex of C that lies on P. It can be shown that in this case the labeling does not depend on the choice of the elementary path. Thus, the labeling of C depends only on Γ, and we omit the superscript P. We are now ready to present our characterization.

Definition 1 *An ortho-radial representation is* valid *if the following conditions hold.*

1. *The sum of angles around each vertex is* 360°.
2. *For each face f, it is*
$$\mathrm{rot}(f) = \begin{cases} 4, & \text{if } f \text{ is a regular face} \\ 0, & \text{if } f \text{ is the central/outer face.} \end{cases}$$
3. *For each simple essential cycle C in G, it is $\ell_C(e) = 0$ for all edges e of C, or there are edges e_+ and e_- on C with $\ell_C(e_+) > 0$ and $\ell_C(e_-) < 0$.*

We have already seen that the first two conditions are necessary. The last condition is new and guarantees that all cycles in the graph can be drawn consistently. For an essential cycle C that violates condition 3 either all labels of edges on C are non-negative or all are non-positive. Then C is called *decreasing* and *increasing*, respectively. Both increasing and decreasing cycles are called *monotone*. Note that an increasing (decreasing) cycle contains an edge with a negative (positive) label. Cycles with only the label 0 are not monotone. Our main result is as follows.

Theorem 1 *Let G be a 4-plane graph with an ortho-radial representation Γ. Then G has an ortho-radial drawing that corresponds to Γ if and only if Γ is valid.*

3 Rectangular Graphs

Let G be a 4-planar graph and let Γ be an ortho-radial representation of G where every face is rectangular. We use a flow method similar to Tamassia [6]. For each edge e, we find an arbitrary path P from $e^\star$ to e, and we determine e as pointing right, down, left or up, if $\mathrm{rot}(P) \mod 4$ is $0, 1, 2$, or 3, respectively. We note that conditions 1 and 2 of valid ortho-radial representations guarantee that this is well-defined. We then reverse the downward and left edges so that all edges point either up or right. The rectangular property of the faces guarantees that each internal face is bounded by two vertical and by two horizontal paths.

We create a radial flow network N_{rad} with a vertex for each face, and an edge from a face f to a face g if and only if there is a horizontal edge with f to its right and g to its left; see Fig. 2. Similarly, we define a vertical flow network N_{ver} that has a vertex for each internal face and an edge from f to g if and only if there is a vertical edge with f to its left and g to its right. We set the capacities of all edges to ∞ and require a minimum flow of 1 on each edge. It is then readily seen that drawings of Γ correspond bijectively to pairs $(F_{\text{rad}}, F_{\text{ver}})$ where F_{rad} is a flow from the central face to the outer face in N_{rad} and F_{ver} is a circulation in N_{ver}. The fact that such a flow exists in N_{rad} is analogous to the orthogonal case [6].

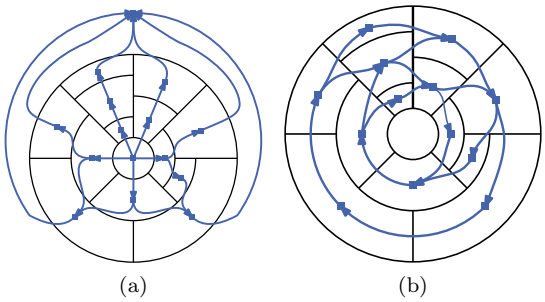
The key is to show that a circulation in N_{ver} exists if Γ is valid. The main idea is to determine for each arc a of N_{ver} a cycle C_a in N_{ver} that contains a. If F_a denotes the circulation that pushes one unit of flow along the arcs of C_a and is 0 elsewhere, then $F_{\text{ver}} = \sum_{a \in A} F_a$, where A denotes the arc set of N_{ver}, is the desired flow. The only reason why such a cycle might not exist is if there is a set S of vertices in N_{ver} such that there exists an arc entering S but no arc exiting S. Without loss of generality, we assume $N_{\text{ver}}[S]$ is weakly connected, which implies that S corresponds to a connected set $\mathcal{S}$ of faces in G. Note that S contains a directed cycle of N_{ver}, which is an essential cycle. Let C and C' denote the smallest and largest essential cycle of G, respectively, such that all faces in $\mathcal{S}$ lie in the interior of C and in the exterior of C'. We show that C is increasing or C' is decreasing.

Assume there is an incoming arc a that crosses C (an incoming arc crossing C' is analogous). Since all faces are rectangles, there is an elementary path P from $e^\star$ to a vertex v on G only using right and down edges of G. Thus, if w is the first vertex of C after v, it is $\ell_C(vw) = 0$ if vw is horizontal and $\ell_C(vw) = -1$ if vw points up. Since no edge on C is pointing downward, i.e., its label is congruent to 1 mod 4, and the labels between adjacent edges differ by $-1, 0$, or 1, it follows that $\ell_C(e') \in \{-2, -1, 0\}$ for all edges e' of C, i.e., $\ell_C(e') \leq 0$. However, the edge e corresponding to the incoming arc a of S is pointing upwards, and therefore $\ell_C(e) = -1$. Hence C is increasing.

Theorem 2 *Let (G, Γ) be a rectangular graph and its ortho-radial representation. There exists a bend-free ortho-radial drawing of G respecting Γ if and only if Γ is valid.*

4 General 4-planar Graphs

In this section we present the proof of Theorem 1. Following Tamassia's approach for orthogonal drawings [6], our approach is based on augmenting a graph G and its valid ortho-radial representation with additional edges so that it remains valid and becomes rectangular. Then the claim follows from Theorem 2. For the sake of simplicity, we assume that the outer face and the central face are already bounded by a horizontal cycle, if not, we can simply add these cycles and suitably attach them to our graph.



(a) (b)

Figure 2: Networks N_{rad} (a) and N_{ver} (b) for assigning the lengths of radial and vertical edges, respectively.

A regular face f that is not a rectangle contains a left bend at a vertex on its boundary, and since it contains four more right bends than left bends by cond. 2, the boundary of f actually contains a vertex u that is followed by two right bends; see Fig. 3a. We call this a *U-shape*. Let z be a subdivision vertex on the edge e immediately after the second right bend of the U-shape and consider the graph G' and its representation Γ' obtained by adding the edge uz and setting the angles at u and z in the face left of uz to 90°; see Fig. 3a. Tamassia shows that Γ' is a valid orthogonal representation of G'. Since G' has fewer left bends at internal faces than G, this procedure eventually terminates with a rectangular graph.

In the case of ortho-radial drawings the situation is not so simple, since we additionally have to ensure that the insertion does not create any monotone cycles. However, this case cannot occur if the edge uz is vertical. Namely, if inserting the vertical edge uz created a monotone cycle C' in G', then, instead of the edge uz, one can use the cycle $C' - uz$ and (a part of) the U-shape to find a monotone cycle C in G.

Lemma 3 *Vertical augmentation does not create monotone cycles.*

There are, however, faces that do not have a U-shape whose last segment is horizontal; see Fig. 3b. Fix again a vertex u with a U-shape in face f. Indeed, it can be the case that subdividing the last edge of the U-shape by a vertex z and inserting uz as above creates a monotone cycle. In this case, instead of just considering subdividing the last edge of the U-shape as above, we consider all the *candidate edges* e_i incident to f that are opposite of u in the sense that $\text{rot}(C_f[u, e]) = 2$, where C_f denotes the facial cycle of f. Let $e_1, \ldots, e_k$ denote the candidate edges as they occur clockwise starting from u. We call the

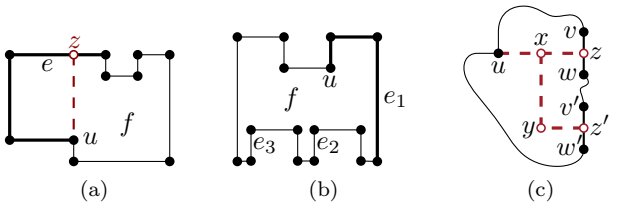


Figure 3: (a) U-shape (thick black) leading to a vertical augmentation (dashed edge). (b) Face f whose U-shapes all require horizontal augmentations and the candidates e_i for u. (c) Structure for simulating the simultaneous augmentation with edges vw and $v'w'$.

process of subdividing e_i with a new vertex z and adding the edge uz *augmenting with* e_i. Intuitively, augmenting with e_i makes a jump further downwards as i increases. Indeed, using similar arguments as the ones for the vertical case, one can show that the first candidate never creates an increasing cycle, and the last candidate never creates a decreasing cycle.

Lemma 4 *Augmenting with e_1 (resp. with e_k) never creates an increasing (resp. decreasing) cycle.*

Moreover, it can be shown that an increasing and a decreasing cycle cannot intersect strictly, and therefore, it is not possible that augmenting with a candidate creates both an increasing and a decreasing cycle. Thus, if each augmentation with respect to the edges e_i yields an increasing or a decreasing cycle, then by Lemma 4, there exists a pair of candidates such that e_i creates a decreasing cycle and e_{i+1} creates an increasing cycle. Let $e_i = vw$ and $e_{i+1} = v'w'$ be directed so that f lies to their right and note that possibly $v' = w$. Let z and z' denote subdivision vertices of e_i and e_{i+1}, respectively. We simulate augmentation with both candidates simultaneously by inserting two vertices x and y as shown in Fig. 3c. By construction, we then find both a decreasing cycle C using the edges ux, xz and a cycle C' using the edges ux, xy, yz' that is increasing except for possibly the edge xy. Since these cycles both contain u but one is decreasing and the other one is increasing, and such cycles cannot strictly intersect, we can infer that, outside of the face f both cycles actually coincide, i.e., their edges all have label 0 outside of f. We thus find a path P in G consisting of only horizontal edges that starts at v' or at w, ends at u and contains all these vertices. Thus, augmenting the graph G by adding the edge from u to the starting point of P creates a cycle consisting of only horizontal edges. It can be argued that such an augmentation is always safe. In fact one can show that no monotone cycle can share a vertex with a horizontal cycle. The following lemma summarizes this discussion.

Lemma 5 *Let f be a face and let u be a vertex on the boundary of f that forms a left bend in f. Let further $e = vw$ and $e' = v'w'$ be two consecutive candidates. If augmentation with e creates an increasing cycle and augmenation with e' creates a decreasing cycle, then augmenting with one of uw or uv' does not create a monotone cycle.*

Altogether, this proves that for each left bend in a regular face f there exists an augmentation such that the resulting graph and ortho-radial representation are still valid. Eventually, we thus arrive at a rectangular graph, and Theorem 2 applies.

5 Conclusion

In this work we considered orthogonal drawings of graphs on cylinders. Our main result is a characterization of the 4-plane graphs that can be drawn bend-free on a cylinder in terms of a ortho-radial representation of such drawings.

While our proof for both the rectangular case and the general case are algorithmic, only the former currently has an efficient implementation, e.g., in terms of a flow algorithm. In contrast, the rectangulation procedure from Section 4 requires checking whether augmentations create monotone cycles. Our most important open problem is whether such a check can be performed in polynomial time.

References

[1] T. Bläsius, M. Krug, I. Rutter, and D. Wagner. Orthogonal graph drawing with flexibility constraints. *Algorithmica*, 68(4):859–885, 2014.

[2] T. Bläsius, S. Lehmann, and I. Rutter. Orthogonal graph drawing with inflexible edges. *Computational Geometry: Theory and Applications*, 55:26–40, 2016.

[3] T. Bläsius, I. Rutter, and D. Wagner. Optimal orthogonal graph drawing with convex bend costs. *Transactions on Algorithms*, 12(3):33, 2016.

[4] M. Hasheminezhad, S.M. Hashemi, B.D. McKay, and M. Tahmasbi. Rectangular-radial drawings of cubic plane graphs. *Computational Geometry: Theory and Applications*, 43:767–780, 2010.

[5] M. Hasheminezhad, S.M. Hashemi, and M. Tahmabasi. Ortho-radial drawings of graphs. *Australasian Journal of Combinatorics*, 44:171–182, 2009.

[6] R. Tamassia. On embedding a graph in the grid with the minimum number of bends. *SIAM Journal on Computing*, 16(3):421–444, 1987.

Aligned Drawings of Planar Graphs

Tamara Mchedlidze* Marcel Radermacher* Ignaz Rutter†

Abstract

Let G be a planar embedded graph and $\mathcal{A}$ be a set of pseudolines passing through G. An *aligned* drawing of G and $\mathcal{A}$ is a planar polyline drawing Γ of G with an arrangement A of lines so that Γ and A have the same topological properties as G and $\mathcal{A}$. We study this problem restricted to two pseudolines. We show that if every edge of a graph is intersected by at most one pseudoline, then the instance has a straight-line aligned drawing. This implies that every configuration of a planar graph with two pseudolines has an aligned drawing with at most one bend. In order to prove this result, we strengthen the result of Da Lozzo et al. [3], and prove that a planar graph G and a pseudoline $\mathcal{C}$ have an aligned drawing with a prescribed convex drawing of the outer face.

1 Introduction

Two fundamental primitives for highlighting structural properties of a graph in a drawing are alignment of vertices such that they are collinear and geometrically separating unrelated graph parts, e.g., making them separable by a straight line. Not surprisingly, both these techniques have been previously considered from a theoretical point of view in the case of planar straight-line drawings.

Da Lozzo et al. [3] study the problem of producing a planar straight-line drawing of a given embedded graph $G = (V,E)$, i.e., G has a fixed combinatorial embedding and outer face, such that a given set $S \subseteq V$ of vertices is collinear. It is clear that if such a drawing exists, then the line containing the vertices in S is a curve starting and ending at infinity that for each edge e of G either fully contains e or intersects e in at most one point, which may be an endpoint. We call such a curve a *pseudoline* with respect to G. Further, the pseudoline contains all the vertices in S. Da Lozzo et al. [3] show that this is a full characterization of the alignment problem, i.e., a straight-line drawing where the vertices in S are collinear exists if and only if there exists a pseudoline $\mathcal{L}$ with respect to G that contains the vertices in S.

Likewise, for the problem of separation, Biedl et al. [1] considered so-called HH-drawings, where given an embedded graph $G = (V,E)$ and a partition $V = A \dot\cup B$, one seeks a planar straight-line drawing of G in which A and B can be separated by a line. Again, it turns out that such a drawing exists if and only if there exists a pseudoline $\mathcal{L}$ with respect to G such that the vertices in A and B are separated by $\mathcal{L}$ in the sense that they are in different *halfplanes* defined by $\mathcal{L}$.

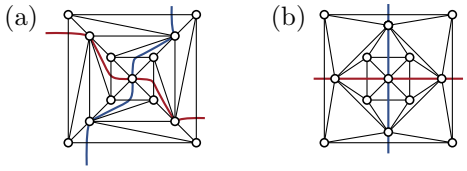


Figure 1: Aligned Drawing (b) of a 2-aligned planar graph (a). The pseudolines $\mathcal{R}$ and $\mathcal{B}$ and the corresponding lines in the drawing are drawn red and blue, respectively.

In particular, the results of Da Lozzo et al. [3] show that given a pseudoline $\mathcal{L}$ with respect to G one can always find a planar straight-line drawing of G such that the vertices on S are collinear and the vertices contained in the halfplanes defined by $\mathcal{L}$ can be separated by a line L. In other words, a topological configuration consisting of a planar graph G and a pseudoline with respect to G can always be stretched. In this paper we initiate the study of this stretchability problem with more than one given pseudoline.

More formally, a tuple $(G,\mathcal{C}_1,\ldots,\mathcal{C}_k)$ is a *k-aligned graph* if $G = (V,E)$ is a planar embedded graph and $\mathcal{C}_1,\ldots,\mathcal{C}_k$ are pseudolines with respect to G. We further require that each pair $\mathcal{C}_i$ and $\mathcal{C}_j$ with $i \neq j$ intersects precisely once. If the number k of curves is clear from the context, we drop it from the notation and simply speak of *aligned graphs*. A tuple $(\Gamma,L_1,\ldots,L_k)$, where each L_i is a line and Γ is a planar drawing of G, is an *aligned drawing of* $(G,\mathcal{C}_1,\ldots,\mathcal{C}_k)$ if and only if the following properties hold; refer to Fig. 1.

1. the arrangement of $L_1,\ldots,L_k$ is isomorphic to the arrangement of $\mathcal{C}_1,\ldots,\mathcal{C}_k$,
2. Γ is homeomorphic to the planar embedding of G,
3. each line L_i intersects in Γ the same vertices and edges as $\mathcal{C}_i$ in G, and it does so in the same order.

We focus on straight-line aligned drawings. For brevity, unless stated otherwise, the term aligned drawing refers to a straight-line drawing throughout this paper.

This convention generalizes the problems studied by Da Lozzo et al. and Biedl et al. who concentrated on the case of a single line. We study a natural extension of their setting and ask for an alignment on two lines. We note that Da Lozzo et al. and Biedl and et al. focus on alignment and separation of given vertices, respectively. Their characteri-

*Department of Computer Science, Karlsruhe Institute of Technology, Germany, radermacher@kit.edu, mched@iti.uka.de

†Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands, i.rutter@tue.nl

zations in terms of existence of pseudolines allows them to abstract from geometry and to construct the pseudolines in a purely combinatorial setting. In contrast to that, we are given multiple pseudolines as part of the input.

In Section 3, we strengthen the result of Da Lozzo et al. and Biedl et al. and show that there exists an aligned drawing of G with a fixed convex drawing of the outer face. In Section 4, we show that every aligned graph $(G,\mathcal{R},\mathcal{B})$ with each edge intersected by at most one pseudoline has an aligned drawing. This immediately implies that every aligned graph $(G,\mathcal{R},\mathcal{B})$ has an aligned drawing with at most one bend.

In addition to the strongly related work mentioned above, there are several other works that are related to the alignment of vertices in drawings. Dujmović [4] shows that every n-vertex planar graph $G=(V,E)$ has a planar straight-line drawing such that $\Omega(\sqrt{n})$ vertices are aligned, and Da Lozzo et al. [3] show that in planar treewidth-3 graphs, one can align $\Theta(n)$ vertices and that in treewidth-k graphs one can align $\Omega(k^2)$ vertices. Chaplik et al. [2] study the problem of covering all edges of a planar graph with a small set of lines. They show that it is $\mathcal{NP}$-hard to decide whether a graph has such a cover of size at most k. The complexity of covering all vertices by at most k lines is open. Dujmović et al. [5] show that there is no set of n lines intersecting in one point that supports all n-vertex planar graphs.

2 Preliminaries and Proof Strategy

Let $G=(V,E)$ be a planar embedded graph with a vertex set V and an edge set E. We call $v\in V$ *interior*, if v does not lie on the boundary of the outer face of G. An edge $e\in E$ is *interior*, if e does not lie entirely on the boundary of the outer face of G. An interior edge is a *chord* if it connects two vertices on the outer face. A point p of an edge e is an *interior* point of e, if p is not an endpoint of e. A *pseudosegment* is a connected bounded subset of a pseudoline. A *triangulation* is a planar graph whose inner faces are all triangles. A *triangulation* of a graph G is a triangulation that contains G as a subgraph. For a graph G and an edge e of G, the graph G/e is obtained from G by contracting e and merging the resulting multiple edges. A *k-wheel* is a wheel graph W_k with k vertices on the outer face and one interior vertex.

Our general strategy for proving the existence of aligned drawings is as follows. First, we show that we can triangulate our instance by adding vertices and edges without invalidating its properties. We can thus assume that our graph G is a triangulation. Second we show that, unless G is very small, e.g., a k-wheel or a triangle, it contains a specific type of edge, namely an edge that is contained in a pseudoline, or an edge that is not intersected by any of the pseudolines. Third, we exploit the existence of such an edge to inductively prove the existence of an aligned drawing of G. Depending on whether the edge is contained in a separating triangle or not, we either decompose along that triangle or contract the edge. In both cases the problem reduces to smaller instances that are almost independent. In order to combine solutions it is, however, crucially important to use the same line R for both of them.

3 Aligned Drawings with One Pseudoline

We show that every aligned graph $(G,\mathcal{R})$ has an aligned drawing (Γ,R). For this extended abstract, we assume that G is 2-connected.

Depending on their relationship to $\mathcal{R}$ the edges of $(G,\mathcal{R})$ are characterized as follows. An edge e of G is *gray* if it does not intersect the curve $\mathcal{R}$. The edge e of G is *red* if it lies entirely on the red curve. Otherwise an edge intersects with $\mathcal{R}$ once and we refer to this edge as *monochromatic*. A vertex on $\mathcal{R}$ is *red* and otherwise *gray*. An *aligned triangulation* of $(G,\mathcal{R})$ is an aligned graph $(G_T,\mathcal{R})$ with G_T a triangulation of G and the same outer face as G.

The following Lemma 1, Lemma 2 and Theorem 3 , in this order, implement the three steps of our proof strategy as outlined above. The proofs are omitted since proofs with similar arguments are given in Section 4.

Lemma 1 *Every aligned graph $(G,\mathcal{R})$ admits an aligned triangulation whose size is linear in G.*

Lemma 2 *Let $(G,\mathcal{R})$ be an aligned triangulation with k vertices on the outer face without a chord. If G is neither a triangle nor a k-wheel, then $(G,\mathcal{R})$ contains a red or a gray edge.*

Proof sketch. Observe the following two properties of aligned triangulation without chords: (i) two red vertices that are consecutive on $\mathcal{R}$ are connected by a red edge, (ii) if G has no red edge, every interior gray vertex is incident to an interior gray edge. With these observations we can show that, if $(G,\mathcal{R})$ does not contain a red or gray edge, there is at most one interior vertex which is red. □

Theorem 3 *Let $(G,\mathcal{R})$ be an aligned triangulation and let (Γ_O,R) be an aligned convex drawing of the aligned outer face $(O,\mathcal{R})$ of G. There exists an aligned drawing (Γ,R) of $(G,\mathcal{R})$ with the same line R and the outer face drawn as Γ_O.*

Given an arbitrary aligned graph $(G,\mathcal{R})$, we can first triangulate it using Lemma 1 and then draw it with Theorem 3. Then the drawing of $(G,\mathcal{R})$ is obtained from this drawing by removing additional vertices and edges.

Corollary 4 *Every aligned graph $(G,\mathcal{R})$ admits an aligned drawing (Γ,R) with a fixed aligned convex drawing (Γ_O,R) of the aligned outer face $(O,\mathcal{R})$.*

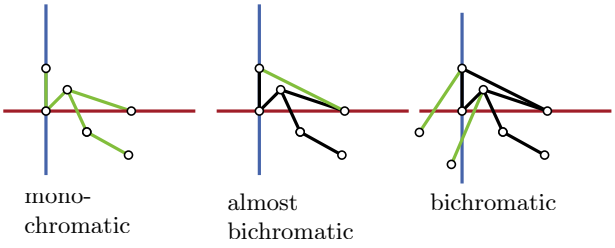


Figure 2: Examples of a monochromatic and a (almost-) bichromatic graph.

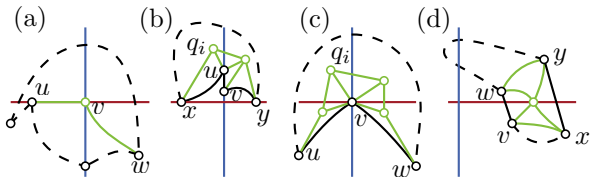


Figure 3: Steps for triangulating almost-bichromatic graphs (black) with monochromatic edges (green).

4 Aligned Drawings with Two Pseudolines

In this section, we show that every aligned graph $(G,\mathcal{R},\mathcal{B})$, where each edge is intersected by at most one of two pseudolines, has an aligned drawing.

The intersection of the four halfplanes defined by $\mathcal{R}$ and $\mathcal{B}$ define four *quadrants*. A vertex or an edge is *red* or *blue* if it lies entirely on $\mathcal{R}$ or $\mathcal{B}$, respectively. A *gray* vertex or edge lies entirely in the interior of a quadrant. An edge is *monochromatic* if it either lies on a pseudoline or shares exactly one point with only one pseudoline, or one of its endpoints lies on the intersection of $\mathcal{R}$ and $\mathcal{B}$. Thus, a monochromatic edge is not necessarily red or blue. *Almost-bichromatic* edges have both endpoints on different pseudolines. Accordingly, a *bichromatic* edge e is always intersected by both pseudolines with at least one intersection point in the interior of e. Every 2-aligned graph is a *bichromatic graph*. An *almost-bichromatic* graph is a bichromatic graph without bichromatic edges. A *monochromatic graph* is an almost-bichromatic graph without almost-bichromatic edges; see Fig. 2.

A *2-aligned triangulation* of $(G,\mathcal{R},\mathcal{B})$ is a 2-aligned graph $(G_T,\mathcal{R},\mathcal{B})$ with G_T a triangulation of G whose outer face is a 4-cycle with each vertex in a different quadrant. We start with the triangulation step of our proof strategy.

Lemma 5 *Let $(G,\mathcal{R},\mathcal{B})$ be a monochromatic aligned graph. There exists a monochromatic aligned triangulation $G_T=(V_T,E_T)$ of G whose size is linear in the size G.*

Proof sketch. Insert a 4-cycle in the unbounded region of G with each vertex in a different quadrant. Since $(G,\mathcal{R},\mathcal{B})$ is monochromatic we can connect each new vertex with a monochromatic or gray edge to the outer face of G.

If f is a non-triangular face whose interior contains the intersection of $\mathcal{R}$ and $\mathcal{B}$, we insert edges uv, vw as shown in Fig. 3(a).

If f is a non-triangular face with a red (blue) edge $e=uv$ we can split f into two faces f' and f'' as shown in Fig. 3(b) such that f' contains e on its boundary. Then we can triangulate f' with monochromatic edges. A similar approach works for red (blue) vertices; see Fig. 3(c).

If f is a non-triangular face whose interior contains a pseudosegment s, then we find two edges vw,xy as shown in Fig. 3(d) and we can triangulate by inserting a vertex on s and monochromatic edges.

If none of the cases above apply, then no non-triangular face contains a part of $\mathcal{R}$ or $\mathcal{B}$. Thus all remaining non-triangular faces can be triangulated with gray edges. □

Next we deal with step two of our proof strategy and show the existence of a specific type of edge unless the instance is very small.

Lemma 6 *Let $(G,\mathcal{R},\mathcal{B})$ be a monochromatic aligned triangulation that does not contain an interior red, blue, or gray edge. Then G is isomorphic to the 4-wheel.*

Proof. Our first goal is to argue that both $\mathcal{R}$ and $\mathcal{B}$ alternately intersect vertices and the interiors of edges of G.

Since $(G,\mathcal{R},\mathcal{B})$ is a monochromatic triangulation, a vertex lies on the intersection of $\mathcal{R}$ and $\mathcal{B}$. As in the proof of Lemma 2 one can argue that if two vertices occur consecutively along $\mathcal{R}$ or $\mathcal{B}$, then we find a red or blue edge, respectively. Now assume that $\mathcal{R}$ intersects two edges e_1,e_2 consecutively. Since G is a triangulation, it follows that e_1 and e_2 share an endpoint x. Moreover, all endpoints of e_1 and e_2 must be gray. Further e_1 and e_2 are consecutive in the circular order of edges around x as otherwise we would either find an intersection with $\mathcal{R}$ between e_1 and e_2 or a gray edge. Thus, e_1 and e_2 bound a face, and hence their endpoints distinct from x are in the same quadrant and connected by an edge e, which is thus gray. Moreover, e cannot be an outer edge as the outer face is a monochromatic 4-cycle and thus does not contain gray edges. It thus follows, that both $\mathcal{R}$ and $\mathcal{B}$ alternate between vertices and edges of G. Moreover, if v is a vertex that is followed by an edge $e=uw$, then u,v,w form a triangle.

Let v be the vertex on the intersection of $\mathcal{R}$ and $\mathcal{B}$. Then there are four triangles $T_1,\dots,T_4$ around v whose edges opposite of v alternately intersect $\mathcal{R}$ and $\mathcal{B}$ in clockwise order. Let u,w be two vertices of these triangles that lie in the same quadrant, without loss of generality, $u\in T_1$ and $w\in T_2$. We show that $u=w$. If not, then let N denote the set of neighbors of v that lie clockwise between u and w together with u and w. Since G is monochromatic, all vertices in N are gray and lie in the same quadrant. Moreover, since G is a triangulation, they form a fan of triangles, and we hence find a gray edge. Since this argument applies to any two consecutive triangles, it follows that the four edges of $T_1,\dots,T_4$ opposite of v form a 4-cycle.

Consider a red (blue) vertex v not lying on the intersection of $\mathcal{R}$ and $\mathcal{B}$. The vertex is incident to two triangles $T_1=(v,x,u),T_2=(v,w,y)$ intersecting the pseudoline $\mathcal{R}$ ($\mathcal{B}$) with x and y in the same quadrant and u and w in the

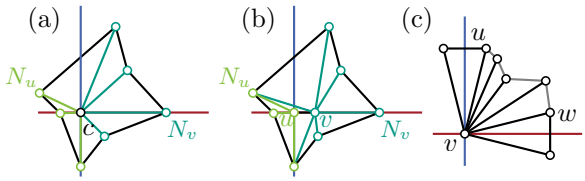


Figure 4: Unpacking an edge in a drawing Γ' of G/e (a) to obtain a drawing Γ of G (b). (c) Fan around vertex v on the intersection of $\mathcal{R}$ and $\mathcal{B}$.

same quadrant. The arguments from above immediately apply to T_1 and T_2, showing that the endpoints u,w and x,y of the triangles T_1,T_2 within the same quadrant are connected by a gray edge, if $u \neq w$ or $x \neq y$ respectively. Since G is a simple graph either $x \neq y$ or $u \neq w$. This concludes the proof. □

The next two lemmas implement step three of our proof strategy and show that the edges that exists by Lemma 6 can be used to reduce the size of the instance. The correctness of Lemma 7 follows from Corollary 4.

Lemma 7 *Let $(G,\mathcal{R},\mathcal{B})$ be a monochromatic aligned triangulation and let T be a separating triangle splitting G into subgraphs $G_{\text{in}},G_{\text{out}}$ such that $G_{\text{in}} \cap G_{\text{out}} = T$ and G_{out} contains the outer face of G. Then $(G_{\text{out}},\mathcal{R},\mathcal{B})$ is a monochromatic aligned triangulation and G has an aligned drawing if and only if G_{out} has an aligned drawing.*

Lemma 8 *Let $(G,\mathcal{R},\mathcal{B})$ be a monochromatic triangulation and let e be a red, blue or gray edge of G that is not contained in a separating triangle. Then the graph $(G/e,\mathcal{R},\mathcal{B})$ is a monochromatic triangulation. Further, $(G,\mathcal{R},\mathcal{B})$ has an aligned drawing if $(G/e,\mathcal{R},\mathcal{B})$ has an aligned drawing.*

Proof. Observe that G/e is simple and triangulated since G does not have separating triangles. Further, if no endpoint of e lies on the intersection of $\mathcal{R}$ and $\mathcal{B}$, G/e remains aligned since e is red, blue or gray, in particular the vertex c resulting from contracting e has the same color as e. If an endpoint v of $e = uv$ lies on the intersection of $\mathcal{R}$ and $\mathcal{B}$ we place the vertex c obtained by contracting e on the intersection as well. Since every neighbor of u keeps its color, G/e remains a monochromatic triangulation.

Let (Γ',R,B) be an aligned drawing of $(G/e,\mathcal{R},\mathcal{B})$. Let Γ'' denote the drawing obtained from Γ' by removing c together with its incident edges and let f denote the face of Γ'' where c used to lie. Since, G/e is triangulated, f is star-shaped and c lies inside the kernel of f; see Fig. 4. We construct a drawing Γ of G as follows. First, we place u at the position of c and insert all edges incident to u; if one of the two vertices u,v lies on the outer face or on the intersection of $\mathcal{R}$ and $\mathcal{B}$, we assume, without loss of generality, that vertex to be u. Note that since G is an aligned triangulation, there is no red or blue edge incident to the intersection of $\mathcal{R}$ and $\mathcal{B}$ or to the outer face. This leaves a uniquely defined inner face f' in which we have to place v. Since G is triangulated, the interior of f' is intersected by at most one pseudoline. Since, we only removed a fan of triangles from f to obtain f', the face f' remains star-shaped and furthermore, all points inside f' sufficiently close to c lie in the kernel of f'. Here it is important that, if the edge is red or blue, the line R or B intersects the kernel of f', respectively. □

Altogether the above lemmas and the observation that an aligned 4-wheel always has an aligned drawing prove our main result.

Theorem 9 *Every monochromatic 2-aligned graph has an aligned drawing.*

By splitting every bichromatic edge with a vertex in the interior of the intermediate quadrant, thus making the graph monochromatic, we get the following corollary.

Corollary 10 *Every bichromatic 2-aligned graph has a polyline aligned drawing with at most one bend per edge.*

5 Conclusion

In this paper we showed that every monochromatic 2-aligned graph $(G,\mathcal{R},\mathcal{B})$ has a straight-line aligned drawing. This immediately implies one-bend aligned drawings of bichromatic aligned graphs. As a tool we showed that an aligned graph $(G,\mathcal{R})$ has an aligned drawing with a fixed convex drawing of the outer face. We conjecture that every almost-bichromatic and possibly also every bichromatic graph $(G,\mathcal{R},\mathcal{B})$ has a straight-line aligned drawing.

References

[1] Biedl, T.C., Kaufmann, M., Mutzel, P.: Drawing planar partitions II: HH-drawings. In: Hromkovič, J., Sýkora, O. (eds.) WG'98. pp. 124–136. Springer (1998)

[2] Chaplick, S., Fleszar, K., Lipp, F., Ravsky, A., Verbitsky, O., Wolff, A.: Drawing graphs on few lines and few planes. In: Hu, Y., Nöllenburg, M. (eds.) GD'16. pp. 166–180. Springer (2016)

[3] Da Lozzo, G., Dujmović, V., Frati, F., Mchedlidze, T., Roselli, V.: Drawing planar graphs with many collinear vertices. In: Hu, Y., Nöllenburg, M. (eds.) GD'16. pp. 152–165. Springer (2016)

[4] Dujmović, V.: The utility of untangling. In: Di Giacomo, E., Lubiw, A. (eds.) GD'15. pp. 321–332. Springer (2015)

[5] Dujmović, V., Evans, W., Kobourov, S., Liotta, G., Weibel, C., Wismath, S.: On graphs supported by line sets. In: Brandes, U., Cornelsen, S. (eds.) GD'10. pp. 177–182. Springer (2011)

On the Relationship between k-Planar and k-Quasi Planar Graphs*

Patrizio Angelini† Michael A. Bekos† Franz J. Brandenburg‡ Giordano Da Lozzo§
Giuseppe Di Battista¶ Walter Didimo‖ Giuseppe Liotta‖ Fabrizio Montecchiani‖
Ignaz Rutter**

Abstract

A graph is k-planar ($k \geq 1$) if it can be drawn (in the plane) such that no edge has more than k crossings. A graph is k-quasi planar ($k \geq 2$) if it can be drawn without k pairwise crossing edges. We prove that, for $k \geq 3$, every k-planar graph is $(k+1)$-quasi planar.

1 Introduction

An emerging research area, informally recognized as *beyond planarity* (see e.g. [6, 8]), concentrates on different models of graph planarity relaxation, which allow edge crossings but forbid specific configurations. Forbidden crossing configurations can be, for example, a single edge that is crossed too many times [9], a group of mutually crossing edges [5, 10], a group of adjacent edges crossed by another edge [4], or an edge that crosses two independent edges [2, 3, 7].

Different models give rise to different families of "beyond planar" graphs. Two of the most popular families introduced in this context are the *k-planar* graphs and the *k-quasi planar* graphs, which are usually defined in terms of *topological graphs*, i.e., graphs with a geometric representation in the plane with vertices as points and edges as Jordan arcs connecting their endpoints. A topological graph is k-planar ($k \geq 1$) if no edge is crossed more than k times, while it is k-quasi planar ($k \geq 2$) if it can be drawn in the plane without k pairwise crossing edges.

A graph is *k-planar* (*k-quasi planar*) if it is isomorphic to some k-planar (k-quasi planar) topological graph. Clearly, k-planar graphs are $(k+1)$-planar and k-quasi planar graphs are $(k+1)$-quasi planar. This naturally defines corresponding hierarchies.

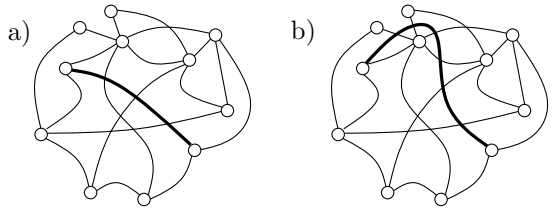


Figure 1: Rerouting the thick edge in the 3-planar graph (a) yields a 4-quasi planar graph (b).

The k-planarity and k-quasi planarity hierarchies have been widely explored in graph theory, graph drawing, and computational geometry, mostly in terms of edge density. While k-planar graphs are known to have at most a linear number of edges [9], the same is not known to be true for k-quasi planar graphs. While linear density upper bounds have been achieved for $k \leq 4$ [1], the best known upper bounds for $k \geq 5$ are super-linear [10]. Despite the fact that both graph hierarchies are well-researched little is known about their relationships.

Contribution. We focus on simple topological graphs and prove the first non-trivial inclusion relationship between the k-planarity and the k-quasi planarity hierarchies. Namely, we show that every k-planar graph is $(k+1)$-quasi planar, for every $k \geq 3$; see Fig. 1.

After some basic terminology in Section 2, Section 3 describes our proof strategy and introduces an edge rerouting technique for removing so-called untangled $(k+1)$-crossings (a $(k+1)$-*crossing* is a set of $(k+1)$ pairwise crossing edges). Section 4 shows that all $(k+1)$-crossings in a k-planar topological graph can be untangled and Section 5 then shows a global rerouting technique to remove all untangled $(k+1)$-crossings.

2 Preliminaries

We only consider graphs with neither parallel edges nor self-loops and without loss of generality we assume all graphs to be connected. We identify the vertices and edges of a topological graph with the points and arcs representing them, respectively. Two edges *cross* if they share one interior point and alternate around this point. Two edges *intersect* if they either cross or share a common endpoint. Graph G is *almost simple*

*The research described in this paper started at the Dagstuhl Seminar 16452 "Beyond-Planar Graphs: Algorithmics and Combinatorics".

†Universität Tübingen, Germany, {angelini,bekos}@informatik.uni-tuebingen.de
‡University of Passau, Germany, brandenb@fim.uni-passau.de
§University of California, Irvine, CA USA, gdalozzo@uci.edu
¶Roma Tre University, Italy, gdb@dia.uniroma3.it
‖Universitá degli Studi di Perugia, Italy, {walter.didimo, giuseppe.liotta, fabrizio.montecchiani}@unipg.it
**TU Eindhoven, The Netherlands, i.rutter@tue.nl

if any two edges cross at most once, and it is *simple* if any two edges intersect at most once. Graph G divides the plane into topologically connected regions, called *faces*. The unbounded region is the *outer face*. Note that the boundary of a face can contain both vertices of the graph and crossing points between edges.

For a subgraph X of a graph G, the *arrangement* $\mathcal{A}_X$ *of* X, is the arrangement of the curves corresponding to the edges of X. We denote the vertices and edges of X by $V(X)$ and $E(X)$. A *node* of $\mathcal{A}_X$ is either a vertex or a crossing point of X. A *segment* of $\mathcal{A}_X$ is a part of an edge of X that connects two nodes.

A k-crossing X is *untangled* if in $\mathcal{A}_X$ all nodes corresponding to vertices in $V(X)$ are incident to a common face, otherwise it is *tangled*; see Sec. 4. A *fan* is a set of edges that share a common endpoint.

Observation 1 *Let $G = (V, E)$ be a k-planar simple topological graph and let X be a $(k+1)$-crossing in G. An edge in $E(X)$ cannot be crossed by any edge in $E \setminus E(X)$. In particular, for any two distinct $(k+1)$-crossings X and Y in G, $E(X) \cap E(Y) = \emptyset$ holds.*

3 Edge Rerouting Operations and Proof Strategy

We introduce an edge rerouting operation that will serve as a basic tool for our proof strategy. Let G be a k-planar simple topological graph and consider an untangled $(k+1)$-crossing X in G; see Fig. 2a.

Let $e = \{u, v\} \in E(X)$ and let $w \in V(X) \setminus \{u, v\}$. Denote by $\mathcal{A}'_X$ the arrangement obtained from $\mathcal{A}_X$ by removing all nodes corresponding to vertices in $V(X) \setminus \{u, v, w\}$, together with their incident segments, and by removing edge (u, v). The operation of *rerouting e around w* consists of redrawing e sufficiently close to the boundary of the outer face of $\mathcal{A}'_X$, choosing the routing that passes close to w, in such a way that e does not cross any edge in $E \setminus E(X)$ except for a fan incident to w; see Fig. 2b. More precisely, let D be a topological disk that encloses all crossing points of X and such that each edge in $E(X)$ crosses the boundary of D exactly twice. Then, the rerouted edge keeps unchanged the parts of e that go from u to the boundary of D and from v to the boundary of D. We call the unchanged parts of a rerouted edge its *tips* and the part that routes around w its *hook*.

Lemma 2 *Let $G' \simeq G$ be the topological graph obtained from G by rerouting an edge $e = \{u, v\} \in E(X)$ around a vertex $w \in V(X) \setminus \{u, v\}$. Let d be the edge of $E(X)$ incident to w. Graph G' has the following properties. (i) Edges e and d do not cross; (ii) The edges that are crossed by e in G' but not in G form a fan at w; (iii) G' is almost simple.*

Note that G' may be non-simple as e may possibly cross its adjacent edges (u, w) and (v, w). We fix this in Section 5 by redrawing (u, w) and (v, w) along e.

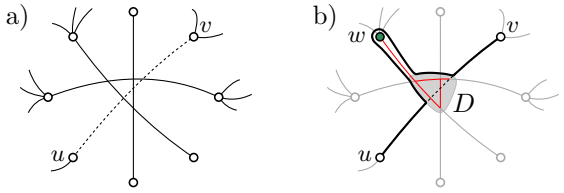


Figure 2: The rerouting operation for dissolving untangled k-crossings. (a) An untangled k-crossing X. (b) The rerouting of the dashed edge (u, v) around the marked vertex w. The arrangement $\mathcal{A}'_X$ is thin red, the removed nodes and segments are gray.

We now describe our strategy for transforming a k-planar simple topological graph G into a simple topological graph $G' \simeq G$ that is $(k+1)$-quasi planar. Note that G is trivially $(k+2)$-quasiplanar, but it may contains $(k+1)$-crossings. The idea is to pick from each $(k+1)$-crossing X in G an edge e_X and a vertex w_X and then to apply the above rerouting operation *simultaneously* for all pairs (e_X, w_X). We call this operation *global rerouting*. Note that this is well-defined due to Observation 1.

There are several constraints that have to be satisfied for such a global rerouting to have the desired effect. First, the rerouting operation works only for untangled $(k+1)$-crossings; we deal with this problem in Section 4. Second, even if all $(k+1)$-crossings are untangled, the graph G' resulting from the global rerouting may be non-simple and/or contain new $(k+1)$-crossings. We overcome these issues in Section 5.

4 Untangling $(k+1)$-Crossings

We show how to untangle $(k+1)$-crossings in k-planar topological graphs. The main idea is as follows. Consider a $(k+1)$-crossing X in a k-planar topological graph G. Since the edges in $E(X)$ already have k crossings, the faces of the arrangement $\mathcal{A}_X$ partition $G - E(X)$ into disjoint subgraphs G_f, one for each face f of $\mathcal{A}_X$; see Fig. 3a. Further, G_f is drawn inside a topological disk D_f whose boundary contains only the vertices in $V(G_f) \cap X$. By suitably deforming and rearranging these subgraphs, e.g., on the outside of a circle, the edges in $E(X)$ can be reinserted so that X either has fewer crossings or is untangled; see Fig. 3b. Note that this does not tangle other $(k+1)$-crossings.

Lemma 3 *Let G be a k-planar simple topological graph. There exists a k-planar simple topological graph $G' \simeq G$ without tangled $(k+1)$-crossings.*

5 Removing Untangled $(k+1)$-Crossings

Let G be a k-planar simple topological graph without tangled $(k+1)$-crossings. First, we show how to remove all $(k+1)$-crossings in G when $k \geq 3$ to obtain a

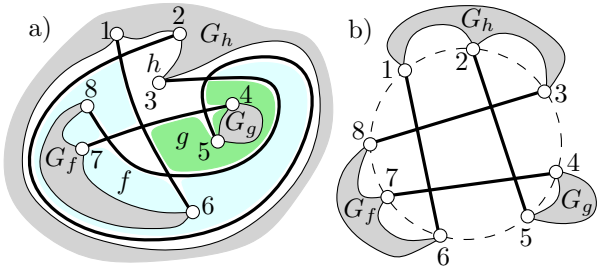


Figure 3: (a) A tangled 4-crossing X in a 3-planar graph G partitions $G - E(X)$ into disjoint subgraphs. (b) Transformation that untangles X.

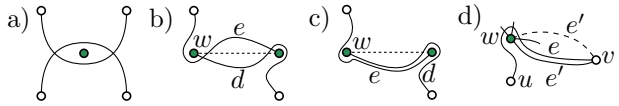


Figure 4: (a–b) Topological graphs that are not almost simple, arising from a global rerouting. (c) Avoiding the non-simplicity in (b) by redrawing one of the two rerouted edges. The vertices used for rerouting are filled green. (d) Illustration for the proof of Lemma 8.

$(k+1)$-quasi planar almost-simple topological graph $G' \simeq G$. Then, we describe how to make this graph simple, without introducing $(k+1)$-crossings.

Let G' be the topological graph obtained from G by performing a global rerouting that picks an edge vertex pair (e_X, w_X) from each $(k+1)$-crossing X in G.

Conditions on the Global Rerouting. We establish conditions on the global rerouting that guarantee that G' is almost simple and $(k+1)$-quasi planar. We start by describing which edge pairs cross newly in G'.

Lemma 4 *If e and d are two edges that cross in G' but not in G, then either both are rerouted around the same vertex, or one of them is rerouted around an endpoint of the other.*

Lemma 4 gives rise to the following criterion for determining whether a global rerouting results in an almost-simple topological graph G'; see Fig. 4a,b.

Lemma 5 *Graph G' is an almost-simple topological graph if and only if the following conditions hold.*
C1. No two edges are rerouted around the same vertex.
C2. There is no pair of edges e, d such that e is rerouted around an endpoint of d and d is rerouted around an endpoint of e.

Moreover, G' is $(k+1)$-quasi planar (though not necessarily almost-simple) if no two edges are rerouted around the same vertex.

Lemma 6 *Let G be a k-planar simple topological graph without tangled $(k+1)$-crossings, and let G' be the graph obtained by a global rerouting operation on G. If G' does not contain two edges rerouted around the same vertex and $k \geq 3$, then G' does not contain any $(k+1)$-crossing.*

Note that Lemma 6 does not hold for $k = 2$.

Obtaining simplicity. Lemmas 3 and 6 imply that, for $k \geq 3$, a given k-planar simple topological graph G can be redrawn such that the resulting topological graph $G' \simeq G$ contains no $(k+1)$-crossings, assuming that no two edges are rerouted around the same vertex. The graph G' may however be not simple, and even not almost simple. We first show how to remove from G' pairs of edges crossing more than once, without introducing any $(k+1)$-crossings. Afterwards we show how to remove crossings between adjacent edges still without introducing any $(k+1)$-crossings.

Lemma 7 *There exists a $(k+1)$-quasi planar almost-simple topological graph G^* such that $G^* \simeq G'$.*

Proof. If G' is not almost simple, then by Lemma 5 there exist pairs of edges such that each of them is rerouted around an endpoint of the other one. We now show how to resolve all such pairs. Let e, d be any of these pairs; see also Fig. 4b. We redraw one of the two edges, say e, sufficiently close along d between the two crossings. More precisely, the tip of e crossed by the hook of d is redrawn following the tip of d crossed by the hook of e, without crossing it (see Fig. 4c). Hence e and d do not cross. We apply this transformation to all such pairs of tips. We claim that the resulting graph G^* does not contain new $(k+1)$-crossings.

Observe first that all such pairs of tips are pairwise disjoint, since no two edges are rerouted around the same vertex. This implies that no tip of an edge is transformed twice in G^*, and that no two transformed edges cross each other. Hence, if a $(k+1)$-crossing exists in G^* then it contains exactly one transformed edge. We prove that this is not the case.

Consider again a pair of edges e, d that cross twice in G' and such that e is transformed in G^*. For an edge l, let X_l denote the $(k+1)$-crossing of G containing l. The edges that cross e are: (i) a set X'_d of edges that cross the tip of d crossed by the hook of e and thus that are part of X_d, (ii) a set X'_e of edges in X_e that cross the tip of e not crossed by d, (iii) a set E_w of edges incident to the vertex w around which e is rerouted (and thus cross the hook of e).

Note that X'_d contains the edges that cross e in G^* and not in G'. These are at most $k-1$ edges. The edges in X'_e do not cross the edges in X'_d, because they are non-rerouted edges that belong to distinct $(k+1)$-crossings of G, and any edge in E_w crosses at most one edge in X'_e. Thus G^* is $(k+1)$-quasi planar.

Finally, we claim that the edges in X'_d are crossed only once by e. Recall that none of these edges crosses

e in G'. Since the tip of e crossed by the hook of d is transformed by following the tip of d crossed by the hook of e, an edge h of X'_d can cross e only once on this tip. On the other hand, it could be that also the other tip of e has been transformed along the tip of an edge l such that it crosses h. But then h crosses tips of two rerouted edges d, l in G', and by Lemma 4 also in G, contradicting the disjointness of X_d and X_l. □

Lemma 8 *There exists a $(k+1)$-quasi planar simple topological graph $\overline{G}$ such that $\overline{G} \simeq G^*$.*

Proof. Since G is simple, if a pair of crossing edges e and e' share an endpoint u, then at least one of them, say e, has been redrawn. Suppose first that only one of them has been redrawn. Under this assumption, we distinguish between two subcases: either e has also been transformed when going from G' to G^* or not.

We first argue about the subcase in which e has not been transformed. Note that, e crosses e' with its hook. Then we redraw e' by following e until reaching u. Now e' crosses only edges that cross the tip of e incident to u. This guarantees that no $(k+1)$-crossing is introduced and that no edge is crossed twice (because G^* is almost simple). See also Fig. 4d.

Consider now the subcase where e has been transformed. Then, there exists an edge e'' that crosses e twice in G', and that has not been transformed. Note that the endpoint around which e'' has been rerouted is not u, as otherwise e' would cross twice e'', which is not possible because G^* is almost simple. Then edge e' is part of the $(k+1)$-crossing of e'' in G. Then we redraw the first part of e from u to the crossing with e' by following e' and leave the rest of e' unchanged. Notice that all the new crossings are due to edges that cross e', and it can be argued that these cannot produce a $(k+1)$-crossing. However, we also need to show that none of these edges is crossed twice. Such an edge would have an endpoint in-between e and e'', which is impossible by the definition of the edge rerouting operation (given that such edge belongs to the same $(k+1)$-crossing of e'' in G).

If both e and e' have been redrawn, then they cannot be redrawn around the same vertex, and the case in which one is redrawn around the non-shared endvertex of the other can be handled similarly as above. □

Existence of global rerouting. It remains to show the existence of a global rerouting where no two edges are rerouted around the same vertex. Consider the bipartite graph H whose nodes correspond to the $(k+1)$-crossings of G and the vertices of G such that $(X, v) \in E(H)$ if and only if $v \in V(X)$. It can be shown that H is bipartite, planar, and each node corresponding to a $(k+1)$-crossing has degree $2k$. It is then not hard to see that Hall's criterion is satisfied and there exists a matching that assigns a vertex to each $(k+1)$-crossing.

Lemma 9 *Let G be a k-planar simple topological graph. There exists a global rerouting on G such that no two edges are rerouted around the same vertex.*

The lemmas from this section imply our main result.

Theorem 10 *Let G be a k-planar simple topological graph for $k \geq 3$, then there exists a $(k+1)$-quasi planar simple topological graph $\overline{G}$ such that $\overline{G} \simeq G$.*

6 Conclusion

We proved that, for any $k \geq 3$, k-planar graphs are $(k+1)$-quasiplanar. Our main open question is whether the same inclusion relation also holds for $k = 2$.

References

[1] E. Ackerman. On the maximum number of edges in topological graphs with no four pairwise crossing edges. *Discrete Comput. Geom.*, 41(3):365–375, 2009.

[2] M.A. Bekos, S. Cornelsen, L. Grilli, S.-H. Hong, and M. Kaufmann. On the recognition of fan-planar and maximal outer-fan-planar graphs. In *GD 2014*, volume 8871 of *LNCS*, pages 198–209. Springer, 2014.

[3] C. Binucci, E. Di Giacomo, W. Didimo, F. Montecchiani, M. Patrignani, A. Symvonis, and I.G. Tollis. Fan-planarity: Properties and complexity. *Theor. Comput. Sci.*, 589:76–86, 2015.

[4] O. Cheong, S. Har-Peled, H. Kim, and H.-S. Kim. On the number of edges of fan-crossing free graphs. *Algorithmica*, 73(4):673–695, 2015.

[5] J. Fox, J. Pach, and A. Suk. The number of edges in k-quasi-planar graphs. *SIAM J. Discrete Math.*, 27(1):550–561, 2013.

[6] M. Kaufmann, S. Kobourov, J. Pach, and S.-H. Hong. Beyond planar graphs: Algorithmics and combinatorics. Dagstuhl Seminar 16452, 2016.

[7] M. Kaufmann and T. Ueckerdt. The density of fan-planar graphs. *CoRR*, abs/1403.6184, 2014.

[8] G. Liotta. Graph drawing beyond planarity: some results and open problems. In *Proc. 15th Italian Conf. Theor. Comput. Sci.*, volume 1231 of *CEUR Workshop Proceedings*, pages 3–8, 2014.

[9] J. Pach and G. Tóth. Graphs drawn with few crossings per edge. *Combinatorica*, 17(3):427–439, 1997.

[10] A. Suk and B. Walczak. New bounds on the maximum number of edges in k-quasi-planar graphs. *Comput. Geom.*, 50:24–33, 2015.

Radial Contour Labeling with Straight Leaders

Benjamin Niedermann* Martin Nöllenburg† Ignaz Rutter‡

Abstract

In this paper we introduce a flexible and general approach for external label placement assuming a given *contour* of the figure prescribing the possible positions of the labels. While much research on external label placement aims for fast labeling procedures for interactive systems, we focus on highest-quality illustrations. We design a new efficient geometric label placement algorithm that is based only on few fundamental design criteria. Yet, other criteria can flexibly be included in the algorithm as hard or soft constraints.

1 Introduction

Atlases of human anatomy play a major role in the education of medical students and the teaching of medical terminology. Such books contain a broad spectrum of filigree and detailed drawings of the human anatomy from different cutaway views. For example, the third volume of the popular human anatomy atlas Sobotta [8] contains about 1200 figures on 384 pages. Figure 1 (labels added by our algorithm) is one of them showing a cross section of the human skull. The usefulness of the figures essentially relies on the naming of the illustrated components. In order not to spoil the readability of the figure by occluding it with text, the names are placed around the figure without overlapping it. Thin black lines, called *leaders*, connecting the features with their names accordingly guarantee that the reader can match names and features correctly. Following preceding research, we call this labeling technique *external label placement*. In this paper we present a flexible and versatile approach for external label placement in figures. We use medical drawings as running example, but occlusion-free label placements are also indispensable for the readability of other highly detailed figures as they occur, for example, in scientific publications, mechanical engineering and maintenance manuals.

Our approach bridges the gap between practical and theoretical results. While previous practical results (e.g., [1,5]) aim for fast approaches using heuristic multi-criteria optimization, previous theoretical results (e.g., [2, 3]) mostly consider simple models, typically with one optimization criterion, e.g., minimizing the total leader length.

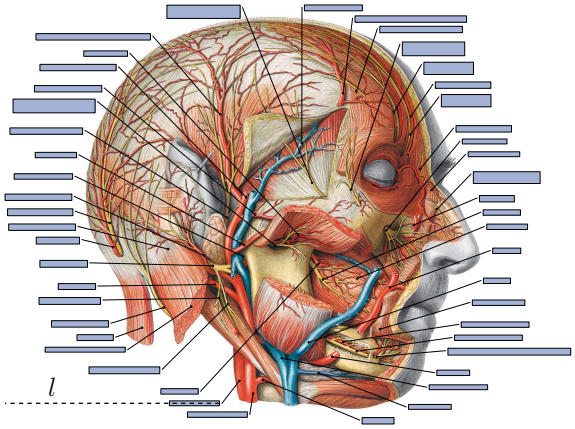


Figure 1: Medical drawing labeled by our approach. Source: Paulsen, Waschke, Sobotta Atlas Anatomie des Menschen, 23.Auflage 2010 ©Elsevier GmbH, Urban & Fischer, München.

Like many of the theoretical results, our approach uses a clear mathematical model to guarantee compliance with pre-defined design rules. However, in contrast to preceding research our approach is significantly more flexible and stands out by its ability to support an easy integration of specific design rules. It particularly relies on only a few key assumptions that most figures with external label placement have in common. Other rules can easily be patched in both as *hard* and *soft* constraints, where hard constraints may not be violated and the compliance of soft constraints is rated by a cost function.

Moreover, in contrast to previous work, our approach also takes costs of consecutively placed labels into account. At first glance this seems to be a small improvement, but in fact it is important to obtain an appealing labeling where, for example, labels have regular distances or the angles of consecutive labels should be similar. Further, our approach supports labels of different sizes. Indeed, for each point feature, the user can pre-define a set of different label sizes modeling formatting rules. The approach also allows to pre-define groups of labels that are placed consecutively, which is required when naming semantically related features.

We first introduce a flexible formal model for *contour labeling*, which is a generalization of boundary

*University of Bonn, Germany
†TU Wien, Austria
‡TU Eindhoven, The Netherlands

labeling (Sect. 2). This model is based on interviews with one layout artist and two editors of the human anatomy atlas Sobotta [8]. We further empirically verified the model by a semi-automatic quantitative analysis of 202 figures printed in the Sobotta [8] atlas. A detailed discussion of the interviews and the semi-automatic analysis is found in [7]; in this preprint we focus on the algorithmic core of our approach, which yields the mathematically optimal solution (Sect. 3).

The strength of our approach comes at the cost of a high asymptotic running time of $O(n^8)$, where n describes the complexity of the input instance. Recently, Keil et al. [6] presented a similar general dynamic programming approach for computing an independent set in outerstring graphs, which can be utilized to solve contour labeling in $O(n^6)$ time for a general cost function rating individual labels; however, it cannot take joint costs of two consecutive labels into account. In contrast to Fink and Suri [4] our approach is significantly faster ($O(n^8)$ instead of $O(n^{15})$) and it supports non-uniform labels and more general shapes.

In the full paper [7], we show in a detailed experimental evaluation on a large set of real-world instances that with some engineering we can solve realistically sized instances in adequate time and high layout quality. Considering different speed-up techniques, the variants of our approaches need between 7 seconds and 346 seconds on average. While the slow variants are optimal, the fast variants achieve near-optimal solutions. The domain experts assessed our algorithm to be a tool of great use that could reduce the working load of a designer significantly.

2 Formal Model

We now describe a model for contour labeling. Let $\mathbb{F}$ be a simple polygon that describes the contour of the figure and contains n points to be labeled, which we call *sites*. We denote the set of the sites by $\mathbb{S}$ and assume that the sites are in general position, i.e., no three sites are collinear. For each site $s \in \mathbb{S}$ we describe its *label*[1] ℓ by a rectangle r and an oriented line segment λ that starts at s and ends on the boundary of r. We call λ the *leader* of ℓ, r the *text box* of ℓ, and the endpoint of λ on r the *port* of ℓ. The other endpoint is the site of ℓ.

A set $\mathcal{L}$ of labels over $\mathbb{S}$ is called an *external labeling* of $(\mathbb{F}, \mathbb{S})$, if (1) $|\mathcal{L}| = |\mathbb{S}|$, (2) for each site $s \in \mathbb{S}$ there is exactly one label in $\mathcal{L}$ that belongs to s, and (3) every text box of a label in $\mathcal{L}$ lies outside of $\mathbb{F}$. If no two labels in $\mathcal{L}$ intersect each other, $\mathcal{L}$ is *planar*. Traversing the figure's boundary in clockwise order starting from the boundary's topmost point defines an ordering on the labels; we call this the radial ordering of $\mathcal{L}$ (in case that a leader intersects the figure's boundary multiple times, we regard the intersection point closest to the port). Two labels are *consecutive* in $\mathcal{L}$ if one directly follows the other in the radial ordering.

Let $\mathcal{L}$ be a planar labeling. Let $\ell_1, \ldots, \ell_n$ be the labels of $\mathcal{L}$ in the radial ordering. For simplicity we define $\ell_{n+1} := \ell_1$. The *cost* c of a labeling $\mathcal{L}$ is defined as $c(\mathcal{L}) = \sum_{i=1}^{n} c_1(\ell_i) + c_2(\ell_i, \ell_{i+1})$, where c_1 is a function assigning a cost to a single label ℓ_i and c_2 is a function assigning a cost to two consecutive labels ℓ_i and ℓ_{i+1}. We note that in contrast to previous research the cost function also supports rating two consecutive labels, which is crucial to assess labels in relation to each other. Given the cost function c, the problem ExternalLabeling then asks for a planar labeling $\mathcal{L}$ of $(\mathbb{F}, \mathbb{S})$ that has minimum cost with respect to c, i.e., for any other planar labeling $\mathcal{L}'$ of $(\mathbb{F}, \mathbb{S})$ it holds that $c(\mathcal{L}) \leq c(\mathcal{L}')$.

We consider the special case that the ports of the labels lie on a common *contour* enclosing $\mathbb{F}$. The contour schematizes the shape of the figure with a certain offset and describes the common silhouette formed by the labels. It thus generalizes the typically rectangular figures studied in boundary labeling [2, 3]. We assume that the contour is given as a simple polygon $\mathbb{C}$ enclosing $\mathbb{F}$. An external labeling $\mathcal{L}$ is called a *contour labeling* if for every label of $\mathcal{L}$ its leader lies inside $\mathbb{C}$ and its port lies on the boundary $\partial\mathbb{C}$ of $\mathbb{C}$. Since not every part of $\mathbb{C}$'s boundary may be suitable for the placement of labels, we require that the ports of the labels are contained in a given subset $\mathbb{P} \subseteq \partial\mathbb{C}$ of candidate ports. If $\mathbb{P}$ is finite, the input instance has *fixed ports* and otherwise *sliding ports*.

A tuple $\mathbb{I} = (\mathbb{C}, \mathbb{S}, \mathbb{P})$ is called an *instance* of contour labeling. The *region* of $\mathbb{I}$ is the region enclosed by $\mathbb{C}$. We restrict ourselves to convex contours and clearly separated sites and text boxes, i.e., we assume that the contour $\mathbb{C}$ is convex and no text box of any label intersects the convex hull of $\mathbb{S}$. Further, we are only interested in *staircase labelings*, i.e., for each label ℓ there is a horizontal half-line l that emanates from the port of ℓ through the text box of ℓ such that no other label intersects l; see Fig. 1. In the full version [7] of this paper, we give empirical evidence that these assumptions are reasonable.

Given a cost function c, the problem ContourLabeling then asks for a (cost) optimal, planar staircase contour labeling $\mathcal{L}$ of $(\mathbb{C}, \mathbb{S}, \mathbb{P})$ with respect to c, i.e., for any other planar staircase contour labeling $\mathcal{L}'$ of $(\mathbb{C}, \mathbb{S}, \mathbb{P})$ it holds that $c(\mathcal{L}) \leq c(\mathcal{L}')$.

3 Algorithm

In this section we describe how to construct the optimal labeling $\mathcal{L}$ of a given instance $(\mathbb{C}, \mathbb{S}, \mathbb{P})$ with respect to a given cost function c. To that end we apply a dynamic programming approach. The ba-

[1] To ease presentation we define that the leader is a component of the label. In preceding research only the rectangle r is called label.

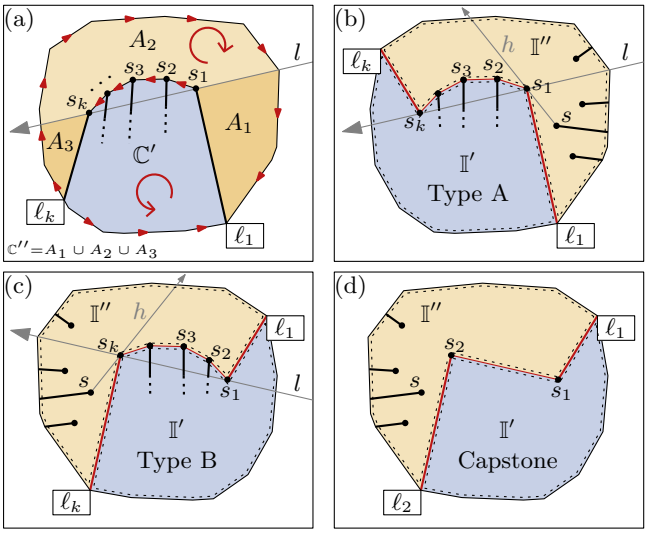


Figure 2: Decomposition in convex instance $\mathbb{I}'$ (blue) and concave instance $\mathbb{I}''$ (orange). (a) Basic definitions. (b) Type A instance with $k > 2$. (c) Type B instance with $k > 2$. (d) Capstone instance.

sic idea is that any optimal contour labeling can be recursively decomposed into a set of sub-labelings inducing disjoint sub-instances. As we show later, these sub-instances are of a special form; we call them *convex sub-instances*. We further show that any such sub-instance can be described by a constant number of parameters over $\mathbb{S}$ and $\mathbb{P}$. Hence, enumerating all choices of these parameters, we enumerate in polynomial time all possible convex sub-instances that an optimal labeling may consist of. For each such sub-instance we compute the cost of an optimal labeling reusing the results of already computed values of smaller sub-instances. In this way we obtain the value of the optimal labeling for $(\mathbb{C}, \mathbb{S}, \mathbb{P})$.

We now sketch the decomposition of a planar labeling $\mathcal{L}$ into a finite set of sub-instances of three types. We describe a sub-instance by a simple polygon that consists of two polylines. One polyline is part of the original contour $\mathbb{C}$ and the other polyline consists of a convex chain of sites and two leaders; see Fig. 2(a). More precisely, assume that we are given a convex chain $K = (s_1, \ldots, s_k)$ of sites with $k \geq 2$ and the two non-intersecting labels ℓ_1 and ℓ_k of s_1 and s_k, respectively. The directed polyline $K' = (p_1, s_1, \ldots, s_k, p_k)$ splits the polygon $\mathbb{C}$ into two polygons $\mathbb{C}'$ and $\mathbb{C}''$, where p_1 and p_k are the ports of ℓ_1 and ℓ_k, respectively. We consider the order of the sites such that we meet p_1 before p_k when going along the contour of $\mathbb{C}$ in clockwise-order starting at the top of $\mathbb{C}$. Further, going along K' we denote the sub-polygon to the left of K' by $\mathbb{C}'$ and to the right of K' by $\mathbb{C}''$. With respect to the direction of K', the sub-polygon $\mathbb{C}'$ is counter-clockwise oriented, while $\mathbb{C}''$ is clockwise oriented. Further, $\mathbb{C}''$ contains the top point of $\mathbb{C}$. We define that $\mathbb{C}'$ contains the sites $s_2, \ldots, s_{k-1}$, while $\mathbb{C}''$ does not.

Thus, the polyline K' partitions the instance $(\mathbb{C}, \mathbb{S}, \mathbb{P})$ into two sub-instances $\mathbb{I}' = (\mathbb{C}', \mathbb{S}', \mathbb{P}')$ and $\mathbb{I}'' = (\mathbb{C}'', \mathbb{S}'', \mathbb{P}'')$ such that

(1) $\mathbb{S}' \cup \mathbb{S}'' = \mathbb{S} \setminus \{s_1, s_k\}$ and $\mathbb{P}' \cup \mathbb{P}'' = \mathbb{P} \setminus \{p_1, p_k\}$,
(2) the sites of $\mathbb{S}'$ lie in $\mathbb{C}'$ or on K and the sites of $\mathbb{S}''$ lie in the interior of $\mathbb{C}''$,
(3) the ports of $\mathbb{P}'$ lie on the boundary of $\mathbb{C}'$ and the ports of $\mathbb{P}''$ lie on the boundary of $\mathbb{C}''$.

Note that the sites s_1, s_k and the ports p_1, p_k neither belong to $\mathbb{I}'$ nor to $\mathbb{I}''$, because they are already used by the fixed labels ℓ_1 and ℓ_k. We call (ℓ_1, ℓ_k, K), which defines the polyline K', the *separator* of $\mathbb{C}'$ and $\mathbb{C}''$.

In the following, we only consider sub-instances, in which the convex chain K lies to the right of the line l through s_1 and s_k pointing towards s_k from s_1; we will show that these are sufficient for decomposing any instance. Put differently, the chain K is a convex part of the boundary of $\mathbb{C}'$ and a concave part of the boundary of $\mathbb{C}''$. We call $\mathbb{I}'$ a *convex* sub-instance and $\mathbb{I}''$ a *concave* sub-instance.

The line l splits $\mathbb{C}''$ into three regions A_1, A_2 and A_3; see Fig. 2(a). Let A_2 be the region to the right of l and let A_1 and A_3 be the regions to the left of l such that A_1 is adjacent to the leader of ℓ_1 and A_3 is adjacent to the leader of ℓ_k. Depending on the choice of ℓ_1 and ℓ_k, the regions A_1 and A_3 may or may not exist. We distinguish the following convex instances. A convex instance has type A (type B) if there is a site $s \in A_1$ ($s \in A_3$) such that ℓ_1 (ℓ_k) and the half-line h emanating from s through s_1 (s_k) separates K from the sites in $\mathbb{C}''$; see Fig. 2(b) and Fig. 2(c).

For both types the chain K is uniquely defined by the choice of ℓ_1, ℓ_k and s, because h separates the sites of $\mathbb{I}'$ from the sites of $\mathbb{I}''$. Thus, type A and type B instances are uniquely defined by ℓ_1, ℓ_k and s; we denote these instances by $\mathbb{I}_{\mathrm{A}}[\ell_1, \ell_k, s]$ and $\mathbb{I}_{\mathrm{B}}[\ell_1, \ell_k, s]$, respectively. We call s the *support point* of the instance. In case that $\mathbb{C}''$ is empty, the chain K is already uniquely defined by ℓ_1 and ℓ_k and we write $\mathbb{I}_{\mathrm{A}}[\ell_1, \ell_k, \perp]$ and $\mathbb{I}_{\mathrm{B}}[\ell_1, \ell_k, \perp]$. Hence, we can enumerate all such instances by enumerating all possible triples consisting of two labels and one site. Since each label is defined by one port and one site, we obtain $O(|\mathbb{S}|^3|\mathbb{P}|^2)$ instances in total.

For $k = 2$ the chain consists of the sites s_1 and s_2 and the support point is superfluous; such an instance is solely defined by the labels ℓ_1 and ℓ_2 of s_1 and s_2, respectively. We call these instances *capstone instances* and denote them by $\mathbb{I}_{\mathrm{C}}[\ell_1, \ell_2]$; see Fig. 2(d).

The next lemma implies that any labeling of any instance $\mathbb{I}$ is a type A instance; see [7] for the proof.

Lemma 1 *Let $\mathbb{I}$ be an instance of* ContourLabeling *and let $\mathcal{L}$ be a planar labeling of $\mathbb{I}$. The first leader ℓ and the last leader ℓ' in the radial ordering of*

$\mathcal{L}$ define a type A instance $\mathbb{I}' = \mathbb{I}_{\mathrm{A}}[\ell, \ell', \perp]$ such that the exterior of $\mathbb{I}'$ is empty.

Hence, optimizing over all choices of first and last labels we find a type A instance that corresponds to an optimal labeling. It remains to show how to solve such an instance. To that end we show that any labeling of that instance can be decomposed into type A, type B and capstone instances recursively.

Let $\mathbb{I} = \mathbb{I}_{\mathrm{A}}[\ell_1, \ell_k, s]$ be a type A instance with support point s, and let $\mathcal{L}$ be a planar labeling of $\mathbb{I}$. By the reasoning above this instance implies a unique convex chain $K = (s_1, \ldots, s_k)$ such that s_1 is the site of ℓ_1 and s_k is the site of ℓ_k; see Fig. 3.

First assume that $\mathbb{I}$ is not a capstone instance, i.e., $k > 2$. We show that $\mathcal{L}$ can be partitioned into a type A instance and a capstone instance as shown in Fig. 3.

To see that, let $\ell_2 \in \mathcal{L}$ be the label of s_2. Since ℓ_2 connects two points of $\mathbb{C}$'s boundary, it partitions $\mathbb{I}$ into two sub-instances $\mathbb{I}'$ and $\mathbb{I}''$ with labelings $\mathcal{L}|_{\mathbb{I}'}$ and $\mathcal{L}|_{\mathbb{I}''}$ such that any label of $\mathcal{L} \setminus \{\ell_2\}$ either is contained in $\mathcal{L}|_{\mathbb{I}'}$ or $\mathcal{L}|_{\mathbb{I}''}$. Let $\mathbb{I}'$ be the instance containing s_1 and $\mathbb{I}''$ the other one. Obviously, $\mathbb{I}'$ forms the capstone instance $\mathbb{I}_{\mathrm{C}}[\ell_1, \ell_2]$. We now show that $\mathbb{I}''$ forms the instance $\mathbb{I}'' = \mathbb{I}_{\mathrm{A}}[\ell_2, \ell_k, s_1]$ of type A.

By definition of $\mathbb{I}$ the label ℓ_1 and the half-line h emanating from s through s_1 separate the convex chain K of $\mathbb{I}$ from the sites in the exterior of $\mathbb{I}$. Because of the convexity of K, the half-line h' emanating from s_1 through s_2 and the label ℓ_2 separate the convex chain $K' = (s_2, \ldots, s_k)$ from the sites in the exterior of $\mathbb{I}''$. Hence, $\mathbb{I}'' = \mathbb{I}_{\mathrm{A}}[\ell_2, \ell_k, s_1]$ has type A.

If $\mathbb{I}$ is a capstone instance, i.e., $k = 2$, the labeling can be decomposed into smaller type A, type B and capstone instances using more intricate arguments. For type B instances we can argue symmetrically to type A instances. Further, the costs of $\mathcal{L}$ can be composed by the costs of the constructed sub-labelings. The details are given in [7].

Based on these results the dynamic programming approach works as follows, where an instance is called *valid* if the two labels ℓ_1 and ℓ_k defining the separator do not intersect and comply with the criterion of a staircase labeling.

Step 1. We compute all valid instances of type A and type B, and all valid capstone instances.

Step 2. We compute the optimal costs for all convex sub-instances. Let $\mathbb{I}$ be the currently considered instance of size $i \geq 0$ with separator $(\ell_1, \ell_k, K = (s_1, \cdots, s_k))$, where the *size* of $\mathbb{I}$ is the number of sites contained in $\mathbb{I}$; recall that s_1 and s_k do not belong to $\mathbb{I}$. Considering the instances in non-decreasing order of their sizes, we can assume that we have already computed the optimal costs for all convex instances with size less than i. Hence, we compute the optimal costs of $\mathbb{I}$ by systematically exploring all decompositions of $\mathbb{I}$ into smaller convex instances.

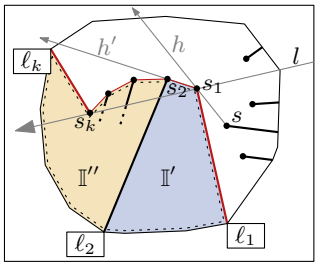


Figure 3: Decomposition of a convex type A instance $\mathbb{I}$ (dashed polygon) into a capstone instance $\mathbb{I}'$ and type A instance $\mathbb{I}''$. The site s is the support point of $\mathbb{I}$ and s_1 is the support point of I''.

Step 3. We explore all choices of first and last labels in the radial ordering. By Lemma 1 one of the choices defines a type A instance that corresponds to an optimal labeling. In the previous steps we have computed the optimal costs for that instance.

In [7] we formally prove that this approach yields a planar staircase labeling in $O(\mathbb{S}^4 \cdot \mathbb{P}^4)$ time.

Theorem 2 ContourLabeling *with fixed ports can be solved in $O(\mathbb{S}^4 \cdot \mathbb{P}^4)$ time.*

References

[1] K. Ali, K. Hartmann, and T. Strothotte. Label layout for interactive 3D illustrations. *Journal of the WSCG*, 13(1):1–8, 2005.

[2] M. A. Bekos, M. Kaufmann, A. Symvonis, and A. Wolff. Boundary labeling: Models and efficient algorithms for rectangular maps. *Computational Geometry: Theory and Applications*, 36(3):215–236, 2007.

[3] M. Benkert, H. J. Haverkort, M. Kroll, and M. Nöllenburg. Algorithms for multi-criteria boundary labeling. *Journal of Graph Algorithms and Applications*, 13(3):289–317, 2009.

[4] M. Fink and S. Suri. Boundary labeling with obstacles. In *Canadian Conf. Computational Geometry (CCCG'16)*, pages 86–92, 2016.

[5] K. Hartmann, K. Ali, and T. Strothotte. Floating labels: Applying dynamic potential fields for label layout. In *Smart Graphics (SG'04)*, volume 3031 of *LNCS*, pages 101–113. Springer, 2004.

[6] J. M. Keil, J. S. B. Mitchell, D. Pradhan, and M. Vatshelle. An algorithm for the maximum weight independent set problem on outerstring graphs. *Computational Geometry: Theory and Applications*, 60:19–25, 2017.

[7] B. Niedermann, M. Nöllenburg, and I. Rutter. Radial contour labeling with straight leaders. *CoRR*, arXiv:1702.01799, 2017.

[8] F. Paulsen and J. Waschke. *Sobotta Atlas of Human Anatomy, Vol. 3, 15th ed., English/Latin: Head, Neck and Neuroanatomy.* Elsevier Health Sciences Germany, 2013.

Formulae Enumerating Polyominoes by both Area and Perimeter*

Gill Barequet† Yufei Zheng†

Abstract

A polyomino is an edge-connected set of cells on $\mathbb{Z}^2$. To-date, no formulae enumerating polyominoes by area (number of cells) or perimeter (number of empty cells neighboring the polyomino) are known. In this paper we present a few formulae enumerating polyominoes according to both parameters.

1 Introduction

A *polyomino* of area n is an edge-connected set of n cells on $\mathbb{Z}^2$. Two (so-called *fixed*) polyominoes are considered equivalent if one can be translated into the other; We consider here only fixed polyominoes. The study of polyominoes began in the 1950s in statistical physics [3], where they are usually called *lattice animals*. In parallel, counting polyominoes has been a long-standing problem in enumerative combinatorics.

Let $A(n)$ denote the number of polyominoes of area n (sequence A001168 in The OEIS [1]). Elements of $A(n)$ are currently known up to $n = 56$ [6]. The asymptotic growth constant of polyominoes has also attracted much attention. Klarner [7] showed that the limit $\lambda := \lim_{n\to\infty} \sqrt[n]{A(n)}$ exists. The convergence of $A(n+1)/A(n)$ to λ (as $n\to\infty$) was proven only three decades later [9]. The best-known lower and upper bounds on λ are 4.0025 [2] and 4.6496 [8], respectively. It is widely *believed* [5] that $\lambda \approx 4.06$.

In statistical physics, the "perimeter" of a polyomino P is defined to be the number of empty cells neighboring cells of P. For example, the perimeter of the polyomino shown in Figure 1 is 12. We denote by $A(n,p)$ the number of polyominoes having area n and perimeter p. Figure 2 shows a plot of the full tabulation of $A(n,p)$ for all $1 \le n \le 26$, in which the curve associated with each value of n is "normalized" by shifting it to the left by $1.195n$, the *empirically*-claimed mean perimeter of all polyominoes of area n [4]. Our goal was to investigate some of the patterns seen in the figure and use them for proving formulae enumerating polyominoes by both area and perimeter.

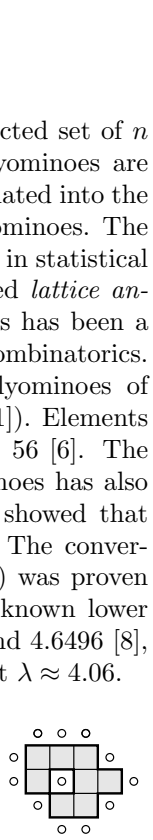


Figure 1: A sample polyomino

*Work on this paper by both authors has been supported in part by ISF Grant 575/15.

†Dept. of Computer Science, The Technion—Israel Institute of Technology, Haifa 32000, Israel. E-mail: {barequet,yufei}@cs.technion.ac.il

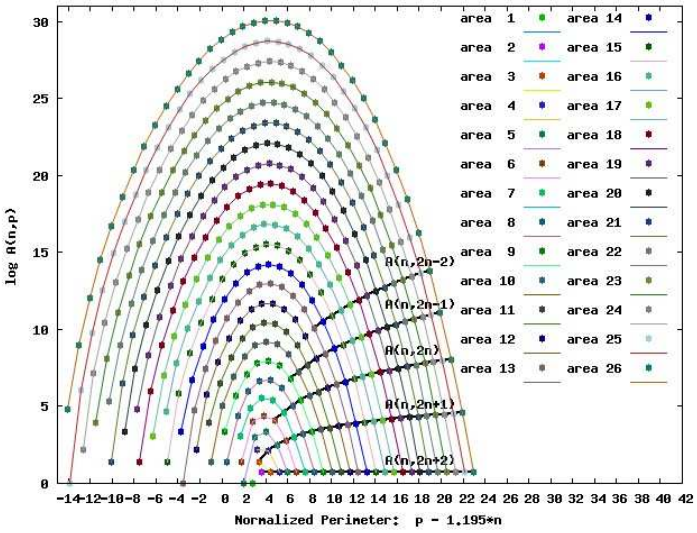


Figure 2: Perimeter distributions

2 Formulae

As shown below, the maximum perimeter of polyominoes of area n is $2n+2$. Hence, we introduce a new parameter, $k \ge 0$, and use the value of k to distinguish between the different cases of perimeter $2n+2-k$.

2.1 Perimeter $2n+2$ ($k=0$)

Observation 1

1. *The maximum possible perimeter of a polyomino of area n is $2n+2$.*
2. $$A(n, 2n+2) = \begin{cases} 1 & n = 1 \\ 2 & n \ge 2 \end{cases}$$

Proof.

1. Consider a polyomino of area n. Each polyomino cell has 4 neighbors, which are either occupied or free. Hence, the total number of cells neighboring polyomino cells is $4n$. However, since the polyomino is edge-connected, there are *at least* $n-1$ neighborhood relations between cells. (The cell-adjacency graphs of polyominoes, that have exactly that number of nodes, are *trees*; Nontree polyominoes have more than $n-1$ cell-neighborhoods.) Each neighborhood reduces the perimeter by 2. Therefore, the maximum possible perimeter is $4n - 2(n-1) = 2n+2$.

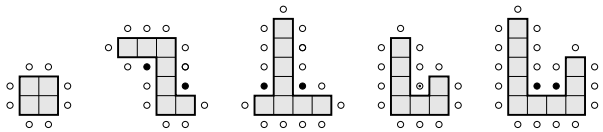


Figure 3: Cases for $k = 2$ (Theorem 1)

2. As above, only tree polyominoes can have the maximum possible perimeter of $2n+2$. However, each "bend" of the form $\square\!\square$ also reduces the perimeter by 1. Therefore, polyominoes with the maximum-possible perimeter cannot have bends. Polyominoes which are trees and have no bends are *sticks*. For area at least 2, there are two sticks (horizontal and vertical). The two "sticks" of area 1 identify into a single cell.

□

The sequence $A(n, 2n+2)|_{n\geq 2} = 2, 2, 2, \ldots$ also has the recursive form $a(n) = a(n-1)$, whose characteristic equation is $x - 1 = 0$.

2.2 Perimeter $2n + 1$ ($k = 1$)

For $n = 1, 2$, the count $A(n, 2n+1) = 0$ holds trivially.

Observation 2 *$A(n, 2n + 1) = 4(n - 2)$ for $n \geq 3$.*

Proof. For a polyomino of area n, the perimeter $2n+1$ is short by only 1 from the maximum possible. This may happen only when the polyomino is a string with a single bend. The bend can be located at any cell in the string except the two extreme cells, and each such shape has four different orientations. □

The sequence $A(n, 2n + 1)|_{n\geq 3} = 4, 8, 12, \ldots$ also has the recursive form $a(n) = 2a(n-1) - a(n-2)$, whose characteristic equation is $x^2 = 2x - 1$, that is, $(x-1)^2 = 0$.

2.3 Perimeter $2n$ ($k = 2$)

In the sequel, we use the following convention in illustrations of polyominoes: Empty circles, filled circles, and "target" circles mark empty cells neighboring one, two, and three cells, respectively, of a polyomino. Reserved cells are marked with crosses.

Theorem 1

$$A(n, 2n) = \begin{cases} 0 & 1 \leq n \leq 3 \\ 9 & n = 4 \\ 28 & n = 5 \\ 6n^2 - 38n + 72 & n \geq 6 \end{cases}$$

Proof. For a polyomino of area n, the perimeter $2n$ is short by 2 from the maximum possible value. Refer to Figure 3. Let us analyze systematically all situations in which this "loss" of two neighbors can happen.

1. Nontrees. Such polyominoes have at least four bends, hence, the loss of at least four neighbors (no matter if the cells inside the corners are occupied or empty). This is true except for the 2×2 polyomino (Figure 3(a)), which has area 4 and perimeter 8. The other cases involve trees.

2. Strings with two bends in opposite directions (Fig. 3(b)). The bends can be located anywhere but the string endpoints, and for each choice of bend locations there are 4 orientations. In total, there are $4\binom{n-2}{2} = 2n^2 - 10n + 12$ possibilities.

3. T-like polyominoes (see Figure 3(c)). Except the junction, there are three non-empty legs whose lengths sum up to $n-1$, hence we need to represent $n-4$ as an ordered sum of three non-negative integers. For each choice of the lengths of the legs, there are four possible orientations. In total, there are $4\binom{n-2}{2} = 2n^2 - 10n + 12$ possibilities.

4. Strings with two bends in the same direction, where the length of the middle leg is 3 and the length of the shorter of the two other legs is 2, see Figure 3(d). (These conditions ensure the loss of two neighbors due to the cell marked with the "target" sign.) For area 5, we have the four orientations of the pentomino $\square\!\square\!\square$. For area at least 6, there are two options for the shorter leg, hence there are 8 possibilities.

5. Strings with two bends in the same direction, and having a middle leg of length at least 4 (see Figure 3(e)). Since the two bends cannot be located at the endpoints of the string, and two cells are reserved in the middle leg, the two bends are chosen out of $n-4$ locations. Finally, each such string has four possible orientations. In total, there are $4\binom{n-4}{2} = 2n^2 - 18n + 40$ possibilities.

To conclude, the formulae enumerating the 5 cases are

Case	Area	Count
1	4	1
2	≥ 4	$2n^2 - 10n + 12$
3	≥ 4	$2n^2 - 10n + 12$
4	5	4
	≥ 6	8
5	≥ 6	$2n^2 - 18n + 40$

Summing up all cases completes the proof. □

The sequence $A(n, 2n)|_{n\geq 6} = 60, 100, 152, \ldots$ also has the recursive form $a(n) = 3a(n-1) - 3a(n-2) + a(n-3)$, whose characteristic equation is $x^3 = 3x^2 - 3x + 1$, that is, $(x-1)^3 = 0$.

2.4 Perimeter $2n-1$ ($k=3$)

Theorem 2

$$A(n,2n{-}1)=\begin{cases}0 & 1\le n\le 4\\ 20 & n=5\\ 80 & n=6\\ 228 & n=7\\ 480 & n=8\\ \left.\begin{matrix}-1053.5-10.5(-1)^n\\ +486.75n+1.25n(-1)^n\\ -89n^2+6.5n^3\end{matrix}\right\} & n\ge 9\end{cases}$$

Proof. Similarly to the proof of Theorem 1, we have that for a polyomino of area n, the perimeter $2n{-}1$ is short by 3 from the maximum possible value. Refer to polyomino types as in Figure 4. Let us explore the situations in which this "loss" of 3 neighbors can happen. In some of the cases, the value of a function $f(n)$ depends on the parity of n: If n is even, then $f(n)=f_1(n)$; otherwise, if n is odd, then $f(n)=f_2(n)$. A simplified form, avoiding a conditional expression, is $f(n)=(f_1(n)+f_2(n))/2+(-1)^n(f_1(n)-f_2(n))/2$.

1. Type (a.1). In this case $a\ge c+1$.
   • n is even. In this subcase, the count is

$$f_1(n)=8\sum_{c=2}^{(n-8)/2}\sum_{a=c+1}^{n-c-7}\sum_{d=1}^{\frac{n-a-c-11/2-(-1)^{a+c}/2}{2}}1$$
$$=(-10+n)(144-34n+2n^2)/6.$$

   The factor 8 is because this shape does not have symmetries. The first summation is over c, the length of the right "leg." Its minimum value is 2 since the two right corners (marked with black circles) are distinct, while its maximum value is $(n-8)/2$: Eight cells are reserved for bends and horizontal legs, and the division by 2 is since the left leg is at least as long as the right leg (plus one cell). The second summation is over a, the length of the left leg. Its minimum value is $c{+}1$ by construction, while its maximum value is $n{-}c{-}7$: Seven cells are reserved for bends and vertical legs. The third summation is over d, the length of the top leg. Its minimum value is 1, while its maximum value is $(n{-}a{-}c{-}11/2{-}(-1)^{a+c}/2)/2$. First, $a{+}c$ cells are reserved for vertical legs. Then, 5 or 6 more cells are reserved, depending on the parity of $a{+}c$. The final division by 2 is since the bottom leg is longer than the top leg.
   • n is odd. In this subcase, the count is

$$f_2(n)=8\sum_{c=2}^{(n-9)/2}\sum_{a=c+1}^{n-c-7}\sum_{d=1}^{\frac{n-a-c-11/2+(-1)^{a+c}/2}{2}}1$$
$$=(-11+n)(126-32n+2n^2)/6,$$

   with a similar reasoning.

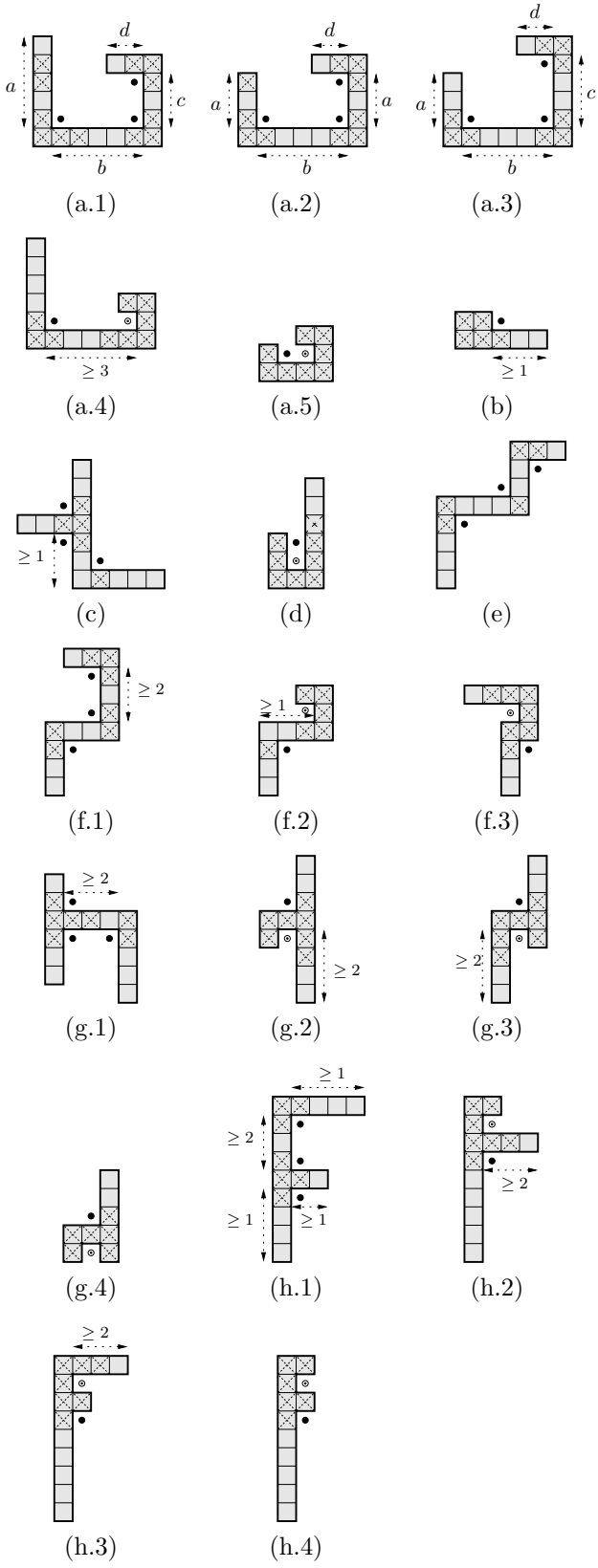


Figure 4: Cases for $k=3$ (Theorem 2)

The total count in this case is

$$\frac{f_1(n)+f_2(n)}{2}+(-1)^n\frac{f_1(n)-f_2(n)}{2}$$
$$=-\frac{471}{2}-\frac{9(-1)^n}{2}+\left(\frac{481}{6}+\frac{(-1)^n}{2}\right)n-9n^2+\frac{n^3}{3}.$$

2. Type (a.2). This case is similar to Case 1 except that $a=c$, hence we have a double instead of a triple summation. We combine the two formulae, for the parities of n, directly into one formula:
$$8\sum_{a=2}^{\frac{n-\frac{1}{2}+\frac{1}{2}(-1)^n-6}{2}}\ \sum_{b=\frac{n-2a-\frac{3}{2}-\frac{1}{2}(-1)^n}{2}}^{n-2a-4}1$$
$$=\frac{111}{2}-\frac{15(-1)^n}{2}+(-15+(-1)^n)\,n+n^2.$$
3. Type (a.3). Here $a<c$ and $d<b$, to distinguish this case from Type (a.1). The analysis is similar.
4. Type (a.4). Since we need to represent $n-9$ as the sum of two non-negative integers, and there are no symmetries, we have $8(n-8)$ possibilities.
5. Type (a.5). This is a special case which exists only for $n=8$. This shape has no symmetries, therefore there are eight possibilities for this case.
6. Type (b). Since this shape does not have any symmetries, there are exactly 8 possibilities.
7. Type (c). This case is similar to other cases (explanations given in the full version of the paper).
8. Type (d). This special case exists only for $n=8$. Polyominoes of this shape have no symmetry, therefore there are eight possibilities for this case.
9. Type (e): strings with 3 bends in alternating directions. Bends can be located anywhere on the string except its endpoints; for each choice of the bends there are 4 orientations. In total, we have $4\binom{n-2}{3}=2n^3/3-6n^2+52n/3-16$ possibilities.

10–16. Types (f.1–f.3, g.1–g.4). These cases are similar to other cases. Explanations are thus omitted here and provided in the full version of the paper.

17. Type (h.1). We need to represent $n-7$ as the ordered sum of four non-negative integers. Since there are no symmetries, there are $8\binom{n-4}{3}=-20+37n/3-5n^2/2+n^3/6$ possibilities.
18. Type (h.2). Aside from reserved cells, we need to represent the number $n-7$ as the ordered sum of two non-negative integers. Since there are no symmetries, this can be done in $8(n-6)$ ways.
19. Type (h.3). This case is identical to Case 18 except that the roles of the two horizontal legs are exchanged. Thus, the number of polyominoes of this type is also $8(n-6)$.
20. Type (h.4). Since there are no degrees of freedom in the lengths of the legs and no symmetries in this type of polyominoes, there are exactly 8 possibilities for this case.

In conclusion, the formulae enumerating all cases are

Case	Area	Count
1	≥ 12	$-\frac{471}{2}-\frac{9(-1)^n}{2}+\left(\frac{481}{6}+\frac{(-1)^n}{2}\right)n-9n^2+\frac{n^3}{3}$
2	≥ 10	$\frac{111}{2}-\frac{15(-1)^n}{2}+(-15+(-1)^n)\,n+n^2$
3	≥ 9	$-\frac{67}{2}+\frac{3(-1)^n}{2}+\left(\frac{211}{12}-\frac{(-1)^n}{4}\right)n-3n^2+\frac{n^3}{6}$
4	≥ 9	$-64+8n$
5	8	8
6	≥ 5	8
7	≥ 5	$-32+\frac{104n}{3}-12n^2+\frac{4n^3}{3}$
8	7	4
	≥ 8	8
9	≥ 5	$-16+\frac{52n}{3}-6n^2+\frac{2n^3}{3}$
10	≥ 7	$-160+\frac{296n}{3}-20n^2+\frac{4n^3}{3}$
11	≥ 6	$-40+8n$
12	≥ 7	$-48+8n$
13	≥ 7	$-160+\frac{296n}{3}-20n^2+\frac{4n^3}{3}$
14	≥ 7	$-48+8n$
15	≥ 7	$-48+8n$
16	≥ 6	8
17	≥ 7	$-160+\frac{296n}{3}-20n^2+\frac{4n^3}{3}$
18	≥ 7	$-48+8n$
19	≥ 7	$-48+8n$
20	≥ 6	8

Summing up all cases completes the proof. The general formula holds already for $n\geq 9$. □

The sequence $A(n,2n-1)|_{n\geq 9}$ obeys the recurrence $a(n)=2a(n-1)+a(n-2)-4a(n-3)+a(n-4)+2a(n-5)-a(n-6)$, with characteristic equation $x^6=2x^5+x^4-4x^3+x^2+2x-1$, i.e, $(x-1)^4(x+1)^2=0$.

2.5 $k=4$ and Beyond

For $k=4$, manual analysis of all cases is not feasible. However, using the known values of $A(5,12)$–$A(26,50)$, we reconstructed the recursive formula for $A(n,2n-2)$ (for $n\geq 13$): $a(n)=2a(n-1)+2a(n-2)-6a(n-3)+6a(n-5)-2a(n-6)-2a(n-7)+a(n-8)$. Its characteristic equation is $x^8=2x^7+2x^6-6x^5+6x^3-2x^2-2x+1$, i.e., $(x-1)^5(x+1)^3=0$.

Future work includes explaining why $A(n,p)$ (n fixed) seems to have a binomial distribution, automating the computation of formulae for $A(n,2n+2-k)$ (k fixed), and understanding their general form.

References

[1] *The On-Line Enc. of Int. Sequences*, `http://oeis.org` .

[2] G. Barequet, G. Rote, and M. Shalah, $\lambda>4$: An improved lower bound on the growth constant of polyominoes, *Comm. of the ACM*, **59**, 88–95, 2016.

[3] S.R. Broadbent and J.M. Hammersley, Percolation processes: I. Crystals and mazes, *Proc. Cambridge Phil. Soc.*, **53**, 629–641, 1957.

[4] A.R. Conway and A.J. Guttmann, On two-dimensional percolation, *J. Phys. A: Math. Gen.*, **28**, 891–904, 1995.

[5] D.S. Gaunt, The critical dimension for lattice animals, *J. of Physics, A: Math. and General*, **13**, L97–L101, 1980.

[6] I. Jensen, Counting polyominoes: A parallel implementation for cluster computing, *Proc. Int. Conf. on Computational Science*, III (Melbourne, Australia and St. Petersburg, Russia, 2003), *LNCS*, 2659, Springer, 203–212.

[7] D.A. Klarner, Cell growth problems, *Canadian J. of Mathematics*, **19**, 851–863, 1967.

[8] D.A. Klarner and R.L. Rivest, A procedure for improving the upper bound for the number of n-ominoes, *Canadian J. of Mathematics*, **25**, 585–602, 1973.

[9] N. Madras, A pattern theorem for lattice clusters, *Annals of Combinatorics*, **3**, 357–384, 1999.

High Dimensional Consistent Digital Segments

Man-Kwun Chiu*,† Matias Korman‡

Abstract

We consider —the problem of digitalizing Euclidean line segments from $\mathbb{R}^d$ to $\mathbb{Z}^d$. Christ *et al.* (DCG, 2012) showed how to construct a set of *consistent digital segments* (CDS) for $d = 2$: a collection of segments connecting any two points in $\mathbb{Z}^2$ that satisfies the natural extension of the Euclidean axioms to $\mathbb{Z}^d$. In this paper we extend this construction to higher dimensions.

We show that any total order can be used to create a set of *consistent digital rays* CDR in $\mathbb{Z}^d$ (a set of rays emanating from a fixed point p that satisfies the extension of the Euclidean axioms), and fully characterize for which total orders we can construct a CDS.

1 Introduction

Computers and digital data have nowadays replaced the ruler and compass methods of computation. In order to have a rigorous system of geometric computation in the digital world, it is desirable to establish a set of axioms similar to those of the Euclidean geometry, where we need to replace a line by a Manhattan path in the micro scale that in a macro scale can be seen as a straight line.

There have been several attempts to define digital segments in a two dimensional $n \times n$ grid. The two dimensional bounded space is the most popular case to consider given its many applications in computer vision and computer graphics. Among the many proposed solutions, we focus on the axiomatic approach introduced by Michael Luby in 1987 [4]. He showed that lines should curve by $\Theta(\log n)$ to satisfy a set of axioms analogous to Euclid's axioms. The theory was recently re-discovered by Chun *et al.* [3] and Christ *et al.* [2], who proposed a d-dimensional version of the set of axioms. Unfortunately, they left open how to generate such a complete set of digital segments that resembles the Euclidean ones. In this paper we provide the first significant step towards answering the question for high dimensions.

*National Institute of Informatics (NII), Tokyo, Japan. chiumk@nii.ac.jp

†JST, ERATO, Kawarabayashi Large Graph Project.

‡Tohoku University, Sendai, Japan. mati@dais.is.tohoku.ac.jp. Partially supported by the ELC project (MEXT KAKENHI No. 12H00855 and 15H02665).

2 Preliminaries

Let $x_1, x_2, \ldots, x_d$ denote the coordinate axes in $\mathbb{Z}^d$, and p_i denote the i-th coordinate of a point $p \in \mathbb{Z}^d$. Our aim is to construct a digital path for any two points $p, q \in \mathbb{Z}^d$ (we denote such a path by $R(p, q)$). Ideally, we want R to be constructive and defined in the whole domain, but sometimes we will consider subsets of $\mathbb{Z}^d \times \mathbb{Z}^d$ instead.

Definition 1 *For any $S \subseteq \mathbb{Z}^d \times \mathbb{Z}^d$, let DS be a set of digital segments such that $R(p, q) \in DS$ for all $(p, q) \in S$. We say that DS forms a* partial set of consistent digital segments *on S (partial CDS for short) if for every pair $(p, q) \in S$ it satisfies the following five axioms:*

(S1) Grid path property: $R(p, q)$ is a path between p and q under the $2d$-neighbor topology[1].

(S2) Symmetry property: $R(p, q) = R(q, p)$.

(S3) Subsegment property: For any $r \in R(p, q)$, we have $R(p, r) \in DS$ and $R(p, r) \subseteq R(p, q)$.

(S4) Prolongation property: There exists $r \in \mathbb{Z}^d$ such that $R(p, r) \in DS$ and $R(p, q) \subset R(p, r)$.

(S5) Monotonicity property: For all $i \leq d$ such that $p_i = q_i$, it holds that every point $r \in R(p, q)$ satisfies $r_i = p_i = q_i$.

A partial CDS is called a set of *consistent digital segments* (CDS for short) when $S = \mathbb{Z}^d \times \mathbb{Z}^d$. A partial CDS is called a *consistent digital ray* system (CDR for short) when $S = \{p\} \times \mathbb{Z}^d$ (for some $p \in \mathbb{Z}^d$). Our aim is to create a CDS in $\mathbb{Z}^d$.

The straightness or resemblance between the digital line segment $R(p, q)$ and the Euclidean segment $\overline{pq}$ is often measured using the *Hausdorff* distance. The Hausdorff distance $H(A, B)$ of two objects A and B is defined by $H(A, B) = \max\{h(A, B), h(B, A)\}$, where $h(A, B) = \max_{a \in A} \min_{b \in B} ||(a, b)||_2$, where $||(a, b)||_2$ is the Euclidean distance between two points.

Definition 2 *Let $DS(S)$ be a partial CDS. We say that $DS(S)$ has Hausdorff distance $f(n)$ if for all $p, q \in S$ such that $||\overline{pq}||_2 \leq n$, $H(\overline{pq}, R(p, q)) = O(f(n))$.*

[1] The $2d$-neighbor topology is the natural one that connects to your predecessor and successor in each dimension. Formally speaking, two points are connected if and only if their L_1 distance is exactly one.

2.1 Previous work

Although the concept of consistent digital segments was first studied by Luby [4], it received renewed interest by the community when it was rediscovered by Chun *et al.* [3]. The latter showed how to construct a set of consistent digital rays (CDR) in any dimension. The construction satisfies all axioms, including the Hausdorff distance bound:

Theorem 1 (Theorem 4.4 of [3], rephrased) *For any $d \geq 2$ and $p \in \mathbb{Z}^d$ we can construct a CDR with $O(\log n)$ Hausdorff distance.*

Håstad [4] and Chun *et al.* [3] showed that any CDR in two dimensions must have $\Omega(\log n)$ Hausdorff distance. Thus $\log n$ is the smallest possible distance one can hope to achieve. This result was generalized by Christ *et al.* [2], who shows a correspondence between CDRs in $\mathbb{Z}^2$ and total orders on the integers. In particular, this correspondence can be used to create a CDS in $\mathbb{Z}^2$ that has $O(\log n)$ Hausdorff distance. Note that the $\Omega(\log n)$ lower bound also holds for CDS, so this result is asymptotically tight.

This answers the question of how well can CDSs approximate Euclidean segments in the two dimensional case. However, the question for higher dimensions remains largely open. Although the method of Christ *et al.* [2] cannot be used to construct CDSs or CDRs in higher dimensions, they show that it can create partial CDSs as follows.

Theorem 2 (Theorem 16 of [2], rephrased) *Let $S = \{(x, y) \colon x_i \geq y_i\} \subset \mathbb{Z}^d \times \mathbb{Z}^d$. We can construct arbitrarily many partial CDSs on S.*

Other than Theorems 1 and 2, little or nothing is known for three or higher dimensions. Up to date, the only CDS known in three or higher dimensions is the naive bounding box approach (described in Section 3) that has $\Omega(n)$ Hausdorff distance. In particular, it still remains open whether one can create a CDS in $\mathbb{Z}^d$ with $o(n)$ Hausdorff distance (for $d > 2$).

Further definitions Given two points $p, q \in \mathbb{Z}^d$ such that $p \neq q$, the *slope* of $R(p, q)$ is the sign vector $\mathtt{t} = (t_1, t_2, \ldots, t_d) \in \{+1, -1\}^d$, where $t_i = +1$ if $p_i \leq q_i$ and is -1 if $p_i \geq q_i$. For simplicity, throughout the paper we refer to *the* slope of $R(p, q)$ (whenever p and q have more than one slope we pick one arbitrarily). Let T be the set containing all 2^d slopes of $\mathbb{Z}^d$.

A *total order* θ of $\mathbb{Z}$ is a binary relation on all pairs of integers. We denote that a is smaller than b with respect to θ by $a \prec_\theta b$. We define three operations on total orders: shift, flip and reverse. The *shift* operation is denoted by $\theta + c$ and is the result of adding a constant value c to each integer without changing their binary relations (that is, $a \prec_\theta b$ if and only if $a + c \prec_{\theta+c} b + c$). Similarly, *flipping* is denoted by $-\theta$ and is the result of changing the sign of all binary relations (that is, $a \prec_\theta b$ if and only if $-a \prec_{-\theta} -b$). The *reverse* operation of θ (denoted by θ^{-1}) is the total order resulting in inverting all relationships (that is, $a \prec_\theta b$ if and only if $b \prec_{\theta^{-1}} a$). Sometimes we will restrict a total order θ to an interval $[a, b]$. We denote this by $\theta[a, b]$.

Throughout the paper we will associate a total order to a point p and a slope $\mathtt{t}$. This will be denoted by $\theta^p_\mathtt{t}$. We will omit the subscript or superscript if it is clear from the context or we use the same total order for all slopes or points, accordingly.

2.2 Overview and paper organization

We study properties that CDRs and CDSs must satisfy in high dimensions (i.e., $d \geq 3$), and show that they behave very differently from the two-dimensional counterparts. In Section 3 we introduce the concept of *axis-order*. Although not needed in two dimensions, it allows us to extend the total order construction of Christ *et al.* to higher dimensions. Given a point $p \in \mathbb{Z}^d$, a total order θ on the integers, and a slope $\mathtt{t}$, we construct a partial CDS which we denote by $TOC(\theta, p, \mathtt{t})$. Specifically, it contains segments having an endpoint p and slope $\mathtt{t}$ (that is, an orthant whose apex is p). In order to create a CDR, we combine 2^d such constructions (one for each slope), and characterize when such an approach works. Recall that T is the set containing all possible slopes of $\mathbb{Z}^d$.

Theorem 3 *For any $d > 2$, point $p \in \mathbb{Z}^d$ and set $\{\theta_\mathtt{t} \colon t \in T\}$ of 2^d total orders, $\bigcup_{\mathtt{t} \in T} TOC(\theta_\mathtt{t}, p, \mathtt{t})$ forms a CDR at p if and only if for any $\mathtt{t}, \mathtt{t}' \in T$ it holds that $\theta_\mathtt{t}[\mathtt{t} \cdot p, \infty) = \theta_{\mathtt{t}'}[\mathtt{t}' \cdot p, \infty) - \mathtt{t}' \cdot p + \mathtt{t} \cdot p$.*

This result highly contrasts with the two dimensional counterpart of Christ *et al.* [2]: in two dimensions we have four different slopes (and thus, four associated quadrants). We can use four different total orders (one for each of the quadrants) and the union will always be a CDR. In higher dimensions this is not true: fixing the total order for a single orthant uniquely determines the behaviour of other orthants. In particular, there is a unique way of completing the partial CDS $TOC(\theta, p, \mathtt{t})$ to a CDR which we denote by $TOC(\theta, p)$.

The next step is to consider the union of several CDRs to obtain a CDS. In Section 4 we characterize for which total orders this is possible.

Theorem 4 *θ is a total order such that $\bigcup_{p \in \mathbb{Z}^d} TOC(\theta, p)$ forms a CDS if and only if $\theta = \theta + 2$ and $\theta = -(\theta + 1)^{-1}$.*

This result also contrasts with the two dimensional case: if we replicate the same construction for all points of $\mathbb{Z}^2$, the result will always be a CDS for any

total order. However, in higher dimensions this does only hold for some total orders.

The main difference between two dimensional and higher dimensional spaces is that the construction for two different slopes has a larger portion in common. In two dimensions, two quadrants share at most a line (whose behaviour is unique because of the monotonicity axiom), but in general orthants may share a subspace of dimension $d-1$. The total orders associated to each orthant must behave similarly within the subspace, which creates some dependency between the total orders. More importantly, each orthant shares subspaces with other orthants, and so on. This cascades creating common dependencies that cycle back to the original orthant and highly constrain the total orders. We refer the reader to [1] for more details.

3 Total order construction in high dimensions

The construction of Christ *et al.*[2] explains how to construct segments of slope $(+1,+1)$ in $\mathbb{Z}^2$ (or equivalently, for points in the first quadrant). The segments of different slopes are obtained via symmetry. In higher dimensions it will be useful to have an explicit way to construct segments of any slope. Thus, we first generalize the method of Christ *et al.* for any orthant.

In order to get an idea of our approach, we first look at the folklore bounding box approach to construct a CDS. When defining the path between p and q, we consider the minimum bounding box formed by the two points. The point with smaller x_1 coordinate will move in the x_1 coordinate until reaching the x_1 coordinate of another point. Afterwards, the one with smaller x_2 coordinate will move in the x_2 coordinate, and so on until the two points meet.

So, if $d=3$, for any segment whose slope is $(+1,+1,+1)$ we first do all the movements in the x_1 coordinate, then x_2 coordinate, and finally in the x_3 coordinate. However, if the segment has slope $(+1,-1,-1)$, then the bounding box CDS will travel first in the x_1 coordinate, then x_3 and finally x_2. Intuitively speaking, even though in both cases we are performing the same steps, the order in which we execute each dimension is slightly different (or equivalently, the total order is being interpreted differently). We model this difference in interpretation through a new concept which we call *axis-order*.

Given a slope $(t_1,t_2,\dots,t_d)$, let $a_1,\dots a_k$ be the indices of the coordinates with positive value in increasing order (that is, $t_i=+1$ if and only if $i=a_j$ for some $j\le k$). Similarly, let $b_1,\dots b_{d-k}$ be the indices of the coordinates with negative value in decreasing order. Then, the *axis-order* of $(t_1,t_2,\dots,t_d)$ is $x_{a_1},x_{a_2},\dots,x_{a_k},x_{b_1},\dots,x_{b_{d-k}}$.

Given a point $p\in\mathbb{Z}^d$, a total order θ and a slope $\mathtt{t}$, we construct the set of rays emanating from p with slope $\mathtt{t}$ and axis order $x_{a_1},x_{a_2},\dots,x_{a_d}$. Define the orthant $\mathcal{O}_{\mathtt{t}}(p)=\{q\in\mathbb{Z}^d\colon \mathtt{t}_i\cdot q_i\ge \mathtt{t}_i\cdot p_i\}$: by definition, the segment from p to any point in $\mathcal{O}_{\mathtt{t}}(p)$ has slope $\mathtt{t}$.

For any point $q\in\mathcal{O}_{\mathtt{t}}(p)$ we construct the segment $R(p,q)$. The path from p to q must do $\mathtt{t}\cdot q-\mathtt{t}\cdot p$ steps, out of which $|p_1-q_1|$ will be in the first coordinate, $|p_2-q_2|$ in the second, and so on. We traverse through intermediate points, each time increasing the inner product with $\mathtt{t}$ by one. At each intermediate point r, we check the position of $\mathtt{t}\cdot r$ in $\theta[\mathtt{t}\cdot p,\mathtt{t}\cdot q-1]$; if it is among the $|p_{a_1}-q_{a_1}|$ smallest elements in $\theta[\mathtt{t}\cdot p,\mathtt{t}\cdot q-1]$ then we move in the x_{a_1} coordinate. Otherwise, if it is among the smallest $|p_{a_1}-q_{a_1}|+|p_{a_2}-q_{a_2}|$ elements we move in x_{a_2}, and so on.

For any point $p\in\mathbb{Z}^d$, slope $\mathtt{t}$, and total order θ, we call the collection of segments $\{R(p,q)\colon q\in\mathcal{O}_{\mathtt{t}}(p)\}$ the *total order* construction of θ (centered at p) for the slope $\mathtt{t}$, and denote it by $TOC(\theta,p,\mathtt{t})$.

Lemma 5 *For any $p\in\mathbb{Z}^d$, slope $\mathtt{t}$ and total order θ, the set of segments in $TOC(\theta,p,\mathtt{t})$ forms a partial CDS on $\{p\}\times\mathcal{O}_{\mathtt{t}}(p)$.*

Theorem 3 stated in Section 2.2 shows the relationship that total orders in different slopes must satisfy in order to create a CDR. Intuitively speaking, this correlation is so strong that choosing one total order effectively fixes the rest. We sketch the proof of one implication of the equivalence.

Lemma 6 (Necessary condition for CDRs) *Let $p\in\mathbb{Z}^d$ and $\{\theta_{\mathtt{t}}\colon t\in T\}$ be a set of 2^d total orders such that $\bigcup_{\mathtt{t}\in T}TOC(\theta_{\mathtt{t}},p,\mathtt{t})$ forms a CDR. Then, for any $\mathtt{t},\mathtt{t}'\in T$, it holds that $\theta_{\mathtt{t}}[\mathtt{t}\cdot p,\infty)=\theta_{\mathtt{t}'}[\mathtt{t}'\cdot p,\infty)-\mathtt{t}'\cdot p+\mathtt{t}\cdot p$.*

Proof. [sketch] We prove the statement by contradiction. That is, assume that there exist two slopes $\mathtt{t},\mathtt{t}'$ such that $v\prec_{\theta_{\mathtt{t}}}v'$ but $v'-\mathtt{t}\cdot p+\mathtt{t}'\cdot p\prec_{\theta_{\mathtt{t}'}}v-\mathtt{t}\cdot p+\mathtt{t}'\cdot p$. Without loss of generality, we can choose $\mathtt{t}$ and $\mathtt{t}'$ so that the corresponding orthants share a plane (pick a sequence of intermediate orthants so that pairwise they do, and look at the first time in which the equality is not satisfied). We pick a point q such that $R(p,q)$ has both slope $\mathtt{t}$ and $\mathtt{t}'$, and look at $R(p,q)$ from both the viewpoints of $TOC(\theta_{\mathtt{t}},p,\mathtt{t})$ and $TOC(\theta_{\mathtt{t}'},p,\mathtt{t}')$.

Along the path $R(p,q)$ we look at two intermediate points r and r'. The main feature of these points is that the behaviour of $R(p,q)$ at those points depends on the positions of v and v' in $\theta_{\mathtt{t}}$ (if we look at it from the viewpoint of $TOC(\theta_{\mathtt{t}},p,\mathtt{t})$). Since $v\prec_{\theta_{\mathtt{t}}}v'$, we can choose q in a way that the path will move in different directions at the two points. Then, we study the same segment from the viewpoint of the other orthant. In this case, the behaviour of the same intermediate points will depend on the positions of $v'-\mathtt{t}\cdot p+\mathtt{t}'\cdot p$ and $v-\mathtt{t}\cdot p+\mathtt{t}'\cdot p$ in the shifted total order instead. Thus, if the relationships are reversed,

the two paths behave differently and in particular we cannot have a CDR. □

For any point p, slope $\mathtt{t}$ and total order θ, there is a unique CDR that can be created in this way and contains $TOC(\theta, p, \mathtt{t})$. Since the choice of slope is not important, let $TOC(\theta, p)$ be the unique CDR that contains $TOC(\theta, p, (+1, \ldots, +1))$.

Corollary 7 *For any $p \in \mathbb{Z}^d$ there exist arbitrarily many CDRs with $O(\log n)$ Hausdorff distance.*

4 Necessary and sufficient conditions for CDSs

Next we focus our attention to constructing CDSs. Christ *et al.* [2] showed that if we apply the same total order construction to all points of $\mathbb{Z}^2$ we get a collection of CDRs whose union is always a CDS. For any total order θ, let $TOC(\theta) = \bigcup_{p \in \mathbb{Z}^d} TOC(\theta, p)$. Unlike the two dimensional case, the construction $TOC(\theta)$ does not always yield a CDS in higher dimensions. Theorem 4 stated in Section 2.2 gives necessary and sufficient conditions that the total order must satisfy.

For any point $p \in \mathbb{Z}^d$ and slope $\mathtt{t}$, let $\theta^p_{\mathtt{t}}$ be the total order associated to point p and slope $\mathtt{t}$ in $TOC(\theta)$.

Theorem 8 *If θ is a total order such that $TOC(\theta)$ forms a CDS, then for any $p \in \mathbb{Z}^d$ and slope $\mathtt{t}$ it holds that $TOC(\theta^p_{\mathtt{t}}, p, \mathtt{t}) = TOC(\theta, p, \mathtt{t})$.*

Thus, even if we in principle would allow different total orders, when creating a CDS in this way we must use the same total order θ for all points and all slopes. Again, this contrasts with the $d = 2$ case where we can combine any two total orders for slopes $(+1, +1)$ and $(+1, -1)$. We now sketch the proof of the first necessary condition of Theorem 4.

Lemma 9 *Let θ be a total order such that $TOC(\theta)$ forms a CDS. Then, $\theta = \theta + 2$.*

Proof. [sketch] Choose an arbitrary $\lambda \in \mathbb{Z}$ and consider the affine plane $\mathcal{H} = \{x_3 = \lambda, x_4 = 0, \ldots, x_d = 0\}$. In this plane we look at the origin $p = (0, 0)$, and points $q = (0, -1)$ and $r = (-1, 0)$ (see Figure 1, left). In particular, we look at the third quadrant (the one with slope $(-1, -1)$): first, from Theorem 3 we know that $\theta^p_{(-1,-1)}$ must coincide with θ (on the interval $[\lambda, \infty)$).

The key property is that both $\theta^q_{(-1,-1)}$ and $\theta^r_{(-1,-1)}$ coincide with $\theta + 2$ on the interval $[\lambda + 1, \infty)$. Another property is that the subtree at q in $TOC(\theta^p_{(-1,-1)}, p, (-1, -1))$ must be part of $TOC(\theta^q_{(-1,-1)}, q, (-1, -1))$. Thus, some inequalities from $\theta^p_{(-1,-1)}$ must be preserved in $\theta^q_{(-1,-1)}$. This also holds for r. Moreover, all paths to p must pass through either q or r, which in particular implies that

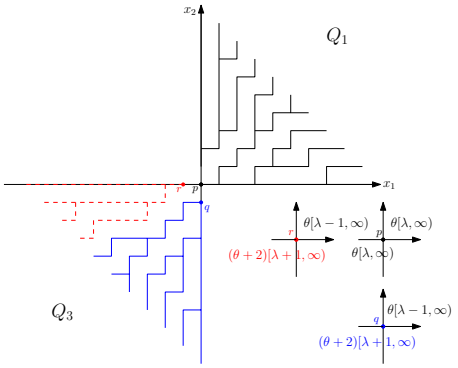


Figure 1: An example of the CDR at p is shown on the left hand side and the relationships between the total orders for the different quadrants at p, q and r. In the example $\theta[0, 8] = \{2 \prec 8 \prec 4 \prec 0 \prec 6 \prec 9 \prec 1 \prec 5 \prec 3 \prec 7\}$.

all inequalities from $\theta^p_{(-1,-1)}$ must also be preserved in either $\theta^q_{(-1,-1)}$ or $\theta^r_{(-1,-1)}$. By combining all of these properties, we show that θ coincides with $\theta + 2$ on the interval $[\lambda + 1, \infty)$. The result works for any value of λ, so when $\lambda \to -\infty$ we get $\theta = \theta + 2$ as claimed. □

Proof. [of Theorem 8] Let $\mathtt{t}' = (+1, \ldots, +1)$ and note that, by definition, we have $\theta^p_{\mathtt{t}'} = \theta$. We apply Theorem 3 and obtain $\theta^p_{\mathtt{t}}[\mathtt{t} \cdot p, \infty) = \theta^p_{\mathtt{t}'}[\mathtt{t}' \cdot p, \infty) - \mathtt{t}' \cdot p + \mathtt{t} \cdot p = \theta[\mathtt{t}' \cdot p, \infty) - \mathtt{t}' \cdot p + \mathtt{t} \cdot p$. The term $-\mathtt{t}' \cdot p + \mathtt{t} \cdot p$ must be an even number (it is the inner product of p with vector $\mathtt{t} - \mathtt{t}'$ which satisfies that each coordinate is either a zero or a two). Thus, we can apply $\theta = \theta + 2$ repeatedly until we get $\theta[\mathtt{t}' \cdot p, \infty) - \mathtt{t}' \cdot p + \mathtt{t} \cdot p = \theta[\mathtt{t} \cdot p, \infty)$ as claimed. □

Let $\mathcal{F}$ be the collection of total orders of $\mathbb{Z}$ that satisfy the conditions of Theorem 4. We bound the Hausdorff distance of the CDSs associated to such total orders.

Theorem 10 *For any $p = (p_1, \ldots, p_d) \in \mathbb{Z}^d$, total order $\theta \in \mathcal{F}$ and $n > 0$, there exists a point $q \in \mathbb{Z}^d$ such that $||p - q||_2 = \Theta(n)$ and $H(\overline{pq}, R(p, q)) = \Theta(n)$.*

References

[1] M.-K. Chiu and M. Korman. High dimensional consistent digital segments. abs/1612.02483, 2016.

[2] T. Christ, D. Pálvölgyi, and M. Stojaković. Consistent digital line segments. *DCG*, 47(4):691–710, 2012.

[3] J. Chun, M. Korman, M. Nöllenburg, and T. Tokuyama. Consistent digital rays. *DCG*, 42(3):359–378, 2009.

[4] M. G. Luby. Grid geometries which preserve properties of Euclidean geometry: A study of graphics line drawing algorithms. In *NATO Conference on Graphics/CAD*, pages 397–432, 1987.

Practical linear-space Approximate Near Neighbors in high dimension

Georgia Avarikioti[1,2], Ioannis Z. Emiris[1], Ioannis Psarros[1], and Georgios Samaras[1]

[1]School of Electrical and Computer Engineering, National Technical U. Athens, Greece, zetavar@hotmail.com
[2]Department of Informatics & Telecommunications, National Kapodistrian University of Athens, Greece, {gsamaras,ipsarros,emiris}@di.uoa.gr

Abstract

The c-approximate Near Neighbor decision problem in high-dimensional spaces has been mainly addressed by Locality Sensitive Hashing (LSH). In practice, however, it is important to ensure linear space usage. Most previous work in this regime focuses on the case that c exceeds 1 by a constant term. We present a simple data structure using linear space and sublinear query time for any $c > 1$: Given an LSH function family for some metric space, we randomly assign a bit to every bucket, eventually projecting points to the Hamming cube of dimension $\leq \lg n$, where n is the number of input points. The search algorithm projects the query, then examines points assigned to nearby vertices on the Hamming cube. We offer an open-source `C++` implementation, and report on several experiments in dimension ≤ 1000 and $n \leq 10^6$. We compared against the state-of-the-art LSH-based library `FALCONN`: our methods are significantly simpler, with comparable performance in terms of query time and storage for one of the LSH families used by `FALCONN`, whereas our code is orders of magnitude faster than brute force.

1 Introduction

We are interested in Approximate Near Neighbor (ANN) search in Euclidean spaces, when the dimension d is high; typically one assumes $d \gg \log n$, where n denotes the number of input datapoints. The (c, r)-ANN problem, where $c > 1$, is defined as follows.

Definition 1 ((c, r)-ANN problem) *Let $(\mathcal{M}, \mathrm{d}_{\mathcal{M}})$ be a metric space. Given $P \subseteq \mathcal{M}$, and reals $r > 0$ and $c > 1$, build a data structure s.t. for any query $q \in \mathcal{M}$, there is an algorithm performing as follows:*

- *if $\exists p^* \in P$ s.t. $\mathrm{d}_{\mathcal{M}}(p^*, q) \leq r$, then return any point $p' \in P$ s.t. $\mathrm{d}_{\mathcal{M}}(p', q) \leq c \cdot r$,*
- *if $\forall p \in P$, $\mathrm{d}_{\mathcal{M}}(p, q) > c \cdot r$, then report "no".*

An important approach for the problem when the dimension is high, is Locality Sensitive Hashing (LSH). The method is based on the idea of using hash functions with the property that it is more probable to map nearby points to the same buckets.

Definition 2 *Let reals $r_1 < r_2$ and $p_1 > p_2 > 0$. We call a family F of hash functions (p_1, p_2, r_1, r_2)-sensitive for a metric space $\mathcal{M}$ if, for any $x, y \in \mathcal{M}$, and h distributed randomly in F, it holds:*

- $\mathrm{d}_{\mathcal{M}}(x, y) \leq r_1 \implies Pr[h(x) = h(y)] \geq p_1$,
- $\mathrm{d}_{\mathcal{M}}(x, y) \geq r_2 \implies Pr[h(x) = h(y)] \leq p_2$.

Let us survey previous work. LSH was introduced by Indyk and Motwani [9] and yields data structures with query time $O(dn^{\rho})$ and space $O(n^{1+\rho} + dn)$. The optimal value of ρ has been extensively studied for several popular metrics, such as ℓ_1 and ℓ_2. In contrast with Definition 2, which concerns data-independent LSH, quite recently the focus has shifted to data-dependent LSH where the algorithms exploit the fact that every dataset has some structure; consequently it improves ρ. Specifically, Andoni and Razenshteyn [5] showed that $\rho = 1/(2c - 1)$ for the ℓ_1 and $\rho = 1/(2c^2 - 1)$ for the ℓ_2 metric. The data-dependent algorithms, though better in theory, are quite challenging in practice. In [3], they present an efficient implementation of one part of [5].

Most of the previous work in the (near) linear space regime focuses on the case that c is greater than 1 by a constant term (e.g. [10]). When $c \to 1^+$, these methods become trivial in the sense that query time becomes linear in n. Two remarkable, recent exceptions are [4] and [7], where they achieve near-linear space and sublinear query time, even for $c \to 1^+$. Their data-dependent data structure is optimal for a reasonable model of hashing-based data structures.

Another line of work yielding linear space and sublinear query is based on random projections to drastically lower-dimensional spaces [1]. The projection ensures that an approximate nearest neighbor in the original space can be found among the preimages of k approximate nearest neighbors in the projection.

In this paper, we specify a random projection from any space endowed with an LSH-able metric. Each LSH function projects points to buckets, each assigned a random bit. This specifies a vertex of the

Hamming hypercube in $\{0,1\}^{\lg n}$, where n is the number of input points. The query algorithm simply projects the query point, then examines points which are assigned to the same or nearby vertices on the Hamming cube. We study standard LSH families for ℓ_2 and ℓ_1, and achieve query time $O(dn^{1-\delta})$, where $\delta = \Theta(\epsilon^2)$, $\epsilon \in (0,1]$. The constants in δ vary with the LSH family, but it holds that $\delta > 0$ for any $\epsilon > 0$. The space and preprocessing time are both linear for constant probability of success, which is important in practical applications.

We illustrate our approach with an open-source implementation, Dolphinn [1] and report on a series of experiments on synthetic and image datasets with $n \leq 10^6$ and $d \leq 1000$. Our algorithm is significantly faster than brute force. Moreover, we handle a real dataset of 10^6 images represented in 960 dimensions with a query time of < 128 msec on average. We compare against LSH-based library `FALCONN` [3], and achieve comparable memory consumption construction time and query time for one of the `FALCONN`'s LSH families, while our algorithm is far simpler (720 lines of code versus 5k).

The rest of the paper is structured as follows. Section 2 states our complexity results and Section 3 presents our experimental results.

2 Data structures

This section introduces our main data structure, and the corresponding algorithmic tools. Given an LSH family of functions for some metric space, we randomly project points to the Hamming cube of dimension $\leq \lg n$ by computing binary strings serving as keys. The query algorithm projects a given point, and tests points assigned to the same or nearby vertices on the hypercube. We start with an ANN data structure whose complexity and performance depends on the LSH family that we assume is available.

Lemma 3 *Given a (p_1, p_2, r, cr)-sensitive hash family F for some metric $(\mathcal{M}, \mathrm{d}_{\mathcal{M}})$ and input $P \subseteq \mathcal{M}$, there exists a data structure for the (c,r)-ANN problem with space $O(dn)$, time preprocessing $O(dn)$, and query time $O(dn^{1-\delta} + n^{H((1-p_1)/2)})$, where*

$$\delta = \delta(p_1, p_2) = \frac{(p_1 - p_2)^2}{(1-p_2)} \cdot \frac{\log \mathrm{e}}{4},$$

where e denotes the basis of the natural logarithm, and $H(\cdot)$ is the binary entropy function. The bounds hold assuming that computing $\mathrm{d}_{\mathcal{M}}(.)$ and the hash function cost $O(d)$. For a given query $q \in \mathcal{M}$, the building process succeeds with constant probability.

[1] `https://github.com/gsamaras/Dolphinn`

Proof. We first sample $h_1 \in F$. We denote by $h_1(P)$ the image of P under h_1. Now for each element $x \in h_1(P)$, with probability $1/2$, set $f_1(x) = 0$, otherwise set $f_1(x) = 1$. This is repeated d' times, and eventually for $p \in \mathcal{M}$ we compute $f(p) = (f_1(h_1(p)), \ldots, f_{d'}(h_{d'}(p)))$. Now, observe that

$$\mathrm{d}_{\mathcal{M}}(p,q) \leq r \implies \mathbb{E}[\|f(p) - f(q)\|_1] \leq 0.5d'(1-p_1),$$

$$\mathrm{d}_{\mathcal{M}}(p,q) \geq cr \implies \mathbb{E}[\|f(p) - f(q)\|_1] \geq 0.5d'(1-p_2).$$

We distinguish two cases.

First, consider the case $\mathrm{d}_{\mathcal{M}}(p,q) \leq r$. Let $\mu = \mathbb{E}[\|f(p) - f(q)\|_1]$. Then, $\Pr[\|f(p) - f(q)\|_1 \geq \mu] \leq 1/2$, since $\|f(p) - f(q)\|_1$ follows the binomial distribution with parameters d' and success probability $(1-p_1)/2$. Second, consider the case $\mathrm{d}_{\mathcal{M}}(p,q) \geq cr$. By typical Chernoff bounds,
$Pr[\|f(p) - f(q)\|_1 \leq exp(-d' \cdot (p_1 - p_2)^2/4(1-p_2))$.

After mapping the query $q \in \mathcal{M}$ to $f(q)$ in the d'-dimensional hamming space we search for all "near" hamming vectors $f(p)$ s.t. $\|f(p) - f(q)\|_1 \leq 0.5d'(1-p_1)$. This search costs $\binom{d'}{1} + \cdots + \binom{d'}{\lfloor d'(1-p_1)/2 \rfloor} \leq O(2^{d' \cdot H((1-p_1)/2)})$. The inequality is obtained from standard bounds on binomial coefficients. Now, the expected number of points $p \in P$ for which $\mathrm{d}_{\mathcal{M}}(p,q) \geq cr$ but are mapped "near" q is $\leq n \cdot exp(-d' \cdot (p_1 - p_2)^2/4(1-p_2)))$. If we set $d' = \lg n$, we obtain expected query time

$$O(n^{H((1-p_1)/2))} + dn^{1-\delta}),$$

where $\delta = \frac{(p_1-p_2)^2}{(1-p_2)} \cdot \frac{\log \mathrm{e}}{4}$. If we stop searching after having seen, say $10n^{1-\delta}$ points for which $\mathrm{d}_{\mathcal{M}}(p,q) \geq cr$, then we obtain the same time with constant probability of success. Notice that "success" translates to successful preprocessing for a fixed query $q \in \mathcal{M}$. The space required is $O(dn)$. □

We set $d' = \lg n$ to minimize the expected number of candidates under the linear space restriction. It is possible to have $d' < \log n$ and sublinear query time.

2.1 The ℓ_2 case

We consider the (c,r)-ANN problem when the dataset is $P \subset \mathbb{R}^d$, and the query is $q \in \mathbb{R}^d$, under the Euclidean metric. We assume, without loss of generality, that $r = 1$, since we can uniformly scale $(\mathbb{R}^d, \|\cdot\|_2)$. We consider a standard LSH family, based on projecting points to random lines, also used in our implementation. Analogous results are obtained [6] if we use the Hyperplane LSH for cosine similarity, by reducing the Euclidean to the spherical problem.

Let p, q be points in $\mathbb{R}^d$ and η their distance. Let $w > 0$ be real, and let t be distributed uniformly in $[0, w]$. In [8], they present the following LSH family. For $p \in \mathbb{R}^d$, consider the random function

$h(p) = \lfloor (\langle p, v\rangle + t)/w \rfloor$, where v is a vector randomly distributed with the d-dimensional normal distribution. For this LSH family, the probability of collIsion is

$$\alpha(\eta, w) = \int_{t=0}^{w} \frac{2}{\sqrt{2\pi}\eta} \exp\Big(-\frac{t^2}{2\eta^2}\Big)\Big(1 - \frac{t}{w}\Big)dt.$$

The implied algorithm suggests the following lemma, whose proof is in [6].

Lemma 4 *Given a set of n points $P \subseteq \mathbb{R}^d$, there exists a data structure for the (c, r)-ANN problem under the Euclidean metric, requiring space $O(dn)$, time preprocessing $O(dn)$, and query time $O(dn^{1-\delta} + n^{0.9})$, where $\delta \geq 0.03\,(c-1)^2$. Given some query $q \in \mathbb{R}^d$, the building process succeeds with constant probability.*

2.2 The ℓ_1 case

We now study the ℓ_1 metric. The dataset is $P \subset \mathbb{R}^d$ and the query $q \in \mathbb{R}^d$. Let us consider the following LSH family [2]:

$$h(p) = \Big(\Big\lfloor \frac{p_1 + t_1}{w} \Big\rfloor, \Big\lfloor \frac{p_2 + t_2}{w} \Big\rfloor, \ldots, \Big\lfloor \frac{p_d + t_d}{w} \Big\rfloor \Big),$$

where $p = (p_1, p_2, \ldots, p_d)$ is a point in P, $w = \alpha r$, and the t_i are drawn uniformly at random from $[0, \ldots, w)$. Buckets correspond to cells of a randomly shifted grid. In order to obtain a better lower bound, we employ an amplified hash function, defined by concatenation of $k = \alpha$ functions $h(\cdot)$ chosen uniformly at random from the above family. This algorithm implies:

Lemma 5 [6] *Given n points $P \subseteq \mathbb{R}^d$, there exists a data structure for the (c, r)-ANN problem under the ℓ_1 metric, requiring space $O(dn)$, time preprocessing $O(dn)$, and query time $O(dn^{1-\delta} + n^{0.91})$, where*

$$\delta \geq 0.05 \cdot \Big(\frac{c-1}{c}\Big)^2.$$

Given some query $q \in \mathbb{R}^d$, the building process succeeds with constant probability.

3 Experimental results

This section presents experiments on our implementation `Dolphinn`, and comparisons on a number of synthetic and real datasets, on a processor at 3 GHz×4 with 8 GB. We compare with the state-of-the-art LSH-based `FALCONN`[2] and its two LSH families, and to the brute force approach.

We use 5 datasets: For special topologies, synthetic *Klein bottle* and *Sphere* sets (n = $10^2 - 10^6$, d = 128 − 1024). The real image datasets are *MNIST* (n = 60k, d = 784)[3], *SIFT* (n = 1m, d = 128), *Small SIFT* (n = 10k, d = 128) and *GIST* (n = 1m, d = 960)[4].

[2] `https://falconn-lib.org/`
[3] `http://yann.lecun.com/exdb/mnist/`
[4] `http://corpus-texmex.irisa.fr/`

n	d	build (sec)	search (μsec)	brute f. (sec)
10^5	128	0.053	62.37	0.006
	256	0.092	152.3	0.012
	512	0.168	257.1	0.025
	800	0.255	374.1	0.039
	1024	0.321	499.6	0.050
100	512	0.0002	1.001	7.5e-05
1000		0.0016	4.924	0.0004
10000		0.0169	47.72	0.0049
100000		0.1683	477.0	0.0499
1000000		1.6800	2529	0.2492

Table 1: Sphere dataset: build, search and brute force, when varying one of n, d.

For the synthetic datasets, we solve the one near neighbor problem with a fixed radius of 1 and we compare to brute force, since `FALCONN` does not provide such a method. For the image datasets, we find all near neighbors within a fixed radius of 1. For fair comparison, both implementations are configured in a way that yields the same accuracy.

The preprocessing time of `Dolphinn` has a linear dependence in n and d, as expected, which is shown in Tables 1 and 2. Moreover, `Dolphinn` is observed to be competitive with `FALCONN` when employing the Cross-Polytope family, but slower than `FALCONN` for the Hyperplane LSH family, as shown in Table 4. Note that any normalization and/or centering of the point set-a requirement for `FALCONN`, is *not* taken into account.

The main issue behind the larger building times is the use of two hash tables which simulates the two random functions, one which maps points to keys in $\mathbb{Z}^{d'}$ and one which maps keys in $\mathbb{Z}^{d'}$ to keys in $\{0,1\}^{d'}$. Using an LSH family which directly maps points to $\{0,1\}^{d'}$ would require only one hashtable, but such an LSH family does not always exist.

We conduct experiments on our synthetic datasets, with n or d fixed. The Sphere dataset is easier than Klein bottle, which explains the reduced accuracy and the decrease of speedup, since more points lie within the fixed radius. Our algorithm scales sublinearly in n and linearly in d, as shown in Tables 1 and 2.

For ANNS, we introduce a threshold on the number of points that the algorithm checks. The search time grows linearly, as expected, while accuracy increases as the threshold increases, as shown in Table 3.

Moreover, we report query times between `FALCONN` and `Dolphinn` on the image datasets; small SIFT, SIFT, MNIST and GIST, for equal memory consumption. `Dolphinn` outperforms the cross-polytope LSH implementation of `FALCONN` and it has comparable performance with the Hyperplane LSH implementation, see Figure 1.

n	d	build (sec)	search (sec)	brute f. (sec)
10^5	128	0.053	0.0009	0.0061
	256	0.091	0.0029	0.0147
	512	0.168	0.0031	0.0254
	800	0.259	0.0056	0.0425
	1024	0.321	0.0061	0.0513
100	512	0.0003	1e-05	7e-05
1000		0.0016	4e-05	0.0004
10000		0.0169	0.0004	0.0049
100000		0.1679	0.0051	0.0501
1000000		1.6816	0.0252	0.2497

Table 2: Klein bottle dataset: build, search and brute force when varying one of n, d.

Thresh.	10^0	10^1	10^2	10^3	10^4	10^5	10^6
search	0.0002	0.0013	0.0139	0.188	2.71	16.7	17.8
accur.	0	0	0	0.26	7.19	80.5	100

Table 3: SIFT dataset: number of points actually checked affects search time (msec) and accuracy (%), i.e. hit if `Dolphinn` answers correctly whether a point lies within the radius or not.

	small SIFT			SIFT			GIST		
d'	4	8	16	4	8	16	4	8	16
`F(c)`	0.01	0.01	0.02	1.07	1.33	2.62	8.21	8.44	18.3
`F(h)`	0.00	0.00	0.00	0.38	0.44	0.72	1.96	2.47	3.66
`D`	0.02	0.02	0.05	0.52	1.49	3.33	4.01	7.98	15.9

Table 4: Build time (sec) for `Dolphinn`, `FALCONN` for Hyperplane/Cross-polytope LSH family. d' is the Hypercube dim. in `Dolphinn`, or #hashbits in `FALCONN`.

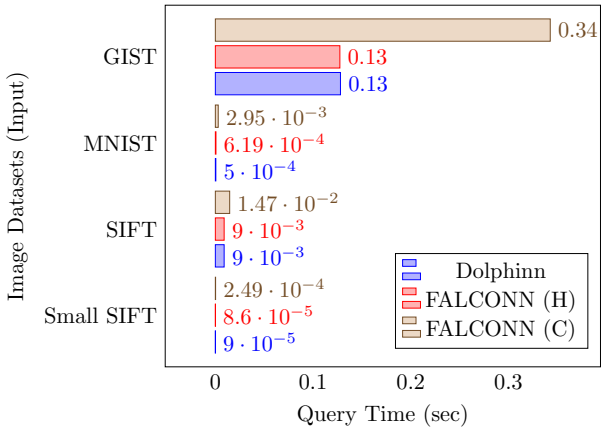


Figure 1: Query time (sec) for `Dolphinn`, `FALCONN` for Hyperplane/Cross-polytope LSH family.

Acknowledgments. All authors partially supported by H2020 MSCA RISE project "Learning and Analysing Massive / Big complex DAta", 2017-2021.

References

[1] E. Anagnostopoulos, I.Z. Emiris, and I. Psarros. Low-quality dimension reduction and high-dimensional approximate nearest neighbor. In *Proc. SoCG*, pages 436–450, 2015. Also: *CoRR*, abs/1412.1683, 2014–2016. Final version under revision for ACM Transactions on Algorithms.

[2] A. Andoni and P. Indyk. Efficient algorithms for substring near neighbor problem. In *Proc. ACM-SIAM SODA*, pages 1203–1212, 2006.

[3] A. Andoni, P. Indyk, T. Laarhoven, I. P. Razenshteyn, and L. Schmidt. Practical and optimal LSH for angular distance. In *Proc. NIPS*, pages 1225–1233, 2015.

[4] A. Andoni, T. Laarhoven, I. Razenshteyn, and E. Waingarten. Optimal hashing-based time-space trade-offs for approximate near neighbors. In *Proc. ACM-SIAM SODA*, 2017.

[5] A. Andoni and I. Razenshteyn. Optimal data-dependent hashing for approximate near neighbors. In *Proc. ACM STOC*, 2015.

[6] G. Avarikioti, I.Z. Emiris, I. Psarros, and G. Samaras. Practical linear-space approximate near neighbors in high dimension. *CoRR*, abs/1612.07405, 2016.

[7] T. Christiani. A framework for similarity search with space-time tradeoffs using locality-sensitive filtering. In *Proc. ACM-SIAM SODA 2017*, pages 31–46, 2017.

[8] M. Datar, N. Immorlica, P. Indyk, and V.S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proc. ACM SoCG*, pages 253–262, 2004.

[9] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proc. STOC*, pages 604–613, 1998.

[10] R. Panigrahy. Entropy based nearest neighbor search in high dimensions. In *Proc. ACM-SIAM SODA*, pages 1186–1195, 2006.

An Experimental Study of Algorithms for Geodesic Shortest Paths in the Constant Workspace Model*

Jonas Cleve† Wolfgang Mulzer†

Abstract

We evaluate experimentally algorithms for finding shortest paths in polygons in the constant workspace model. In this model, the input resides in a read-only array that can be accessed at random. In addition, the algorithm may use a constant number of words for reading and writing. The constant workspace model has been studied extensively in recent years, and algorithms for geodesic shortest paths have received particular attention.

We have implemented three such algorithms and compare them to the classic algorithm by Lee and Preparata that uses linear time and space. We also clarify implementation details that were missing in the original descriptions. Our experiments show that all algorithms perform as advertised and according to the theoretical guarantees. However, the constant factors in the running times turn out to be rather large for the algorithms to be practical.

1 Introduction

In recent years, the *constant workspace model* has enjoyed increasing popularity in computational geometry. Motivated by the increasing deployment of small devices with limited memory capacities, the goal is to develop simple and efficient algorithms for the situation where little workspace is available. The model posits that the input resides in a read-only array that can be accessed at random. In addition, the algorithm may use a constant number of memory words for reading and writing. The output must be written to a write-only memory that cannot be accessed again. Following the initial work from 2011 [2], numerous results have been published for this model, leading to a solid theoretical foundation for dealing with geometric problems when memory is scarce.

But how do these theoretical results measure up in practice? To investigate this question, we have implemented three different algorithms for computing geodesic shortest paths in simple polygons. This is one of the first problems to be studied in the constant workspace model. Given that the general shortest path problem is unlikely to be amenable to constant workspace algorithms, it may be a surprise that a solution for the geodesic case exists at all. By now, several algorithms are known, for constant workspace as well as in the time-space-trade-off regime [1, 8].

Due to the wide variety of approaches and the fundamental nature of the problem, geodesic shortest paths are a natural candidate for an experimental study. Our experiments show that all three algorithms work well in practice and live up to their theoretical guarantees. However, the large running times make them ill-suited for large input sizes.

During our implementation, we also noticed some missing details in the original publications, and we explain below how we have dealt with them.

2 The algorithms

We provide a brief overview over all implemented algorithms; further details can be found in the references. Let P be the input polygon and let $s, t \in P$ be the endpoints of the desired shortest path.

The algorithm by Lee and Preparata. In the classic algorithm, we triangulate P and find the triangles containing s and t. Next, we find the unique path in the dual graph of the triangulation between these two triangles. This gives a sequence $e_1, \ldots, e_m$ of diagonals crossed by the geodesic shortest path. The algorithm walks along these diagonals while maintaining a *funnel*. The funnel has a *cusp* p, initialized to s, and two concave *chains* from p to the two endpoints of the current diagonal e_i. In each step i, there are two cases: (i) if e_{i+1} remains visible from p, we update the appropriate concave chain, using Graham's scan; (ii) if e_{i+1} is not visible from p, we proceed along the appropriate chain until we find the cusp for the next funnel, and we output the vertices encountered along the way as part of the shortest path. Implemented properly, this takes linear time and space [10].

Delaunay. The first constant-workspace-algorithm, called *Delaunay*, directly adapts the method of Lee and Preparata to the constant-workspace model. It was proposed by Asano, Mulzer, and Wang [3] in 2011.

Since we cannot explicitly compute and store a triangulation of P, we use instead a unique implicit triangulation, the *constrained Delaunay triangulation* of P [6]. This triangulation can be navigated efficiently using constant workspace: given a diagonal or a polygon

*Supported by DFG projects MU/3501-1 and RO/2338-6.

†Institut für Informatik, Freie Universität Berlin, Germany. {jonascleve,mulzer}@inf.fu-berlin.de

edge, we can find the incident triangles in $O(n^2)$ time. Using an $O(n)$ time constant-workspace-algorithm for shortest paths in trees, we then enumerate all triangles in the dual graph between the two triangles for s and t in $O(n^3)$ time.

As in Lee and Preparata [10], we need to maintain the visibility funnel while walking along the triangles. Instead of the whole chains, we store only the two line segments that define the current visibility cone, and we recompute the two chains when necessary. The total running time of the algorithm is $O(n^3)$.

Trapezoid. This algorithm was also published by Asano, Mulzer, and Wang [3]. It is based on the same principle as *Delaunay*, but it uses the trapezoidal decomposition of P [5]. Instead of walking along triangles, in $O(n^2)$ time per step, we walk along trapezoids, which takes $O(n)$ time per step. Since there are $O(n)$ steps, the running time improves to $O(n^2)$.

Makestep. This algorithm was presented by Asano et al. [2], and it uses a different approach. We maintain a *current vertex* p of the shortest path together with a *visibility cone*, defined by two points a and b on the boundary of P. The segments pa and pb cut off a subpolygon $P' \subseteq P$. The invariant is that t lies in P'. We gradually shrink P' by advancing a and b, sometimes also relocating p. A charging argument shows that there are $O(n)$ shrinking steps. Each step takes $O(n)$ time, for a total running time of $O(n^2)$.

3 Implementation

We have implemented the algorithms in Python [11]. Graphical output and plots use the `matplotlib` library [9]. Even though there are some geometry packages available for Python, none of them seemed suitable for our needs. Thus, we decided to implement all geometric primitives on our own. The source code of the implementation is available online[1].

For *Lee–Preparata*, we need a triangulation of P. Since polygon triangulation is not the main objective of our study, we relied for this on the *Python Triangle* library by Rufat [12], a wrapper for Shewchuk's *Triangle* [13]. *Triangle* does not provide a linear-time algorithm, but it implements Fortune's sweep, randomized incremental construction, and a divide-and-conquer algorithm, all with a running time of $O(n \log n)$. We used the divide-and conquer algorithm, the default choice. The triangulation phase is not included in the time and memory measurement.

General implementation details. All three constant-workspace algorithms have a general position assumption: *Delaunay* and *Makestep* assume that no three vertices lie on a line. Our implementations also assume general position but throw exceptions if a non-recoverable general position violation is encountered. Most violations, however, can be recovered; e.g. when trying to find the delaunay triangle(s) for a diagonal we can simply ignore points collinear to this diagonal. *Trapezoid* on the other hand assumes that no two vertices have the same x-coordinate. As described by Asano, Mulzer, and Wang [3], this can be fixed by changing the x-coordinate of every vertex to $x + \varepsilon y$ for some small ε such that the x-order of all vertices is maintained. We apply this fix to every polygon in which two vertices share the same x-coordinate.

The coordinates are stored as 64 bit IEEE 754 floats, and the coordinates of randomly generated polygons are rounded to four decimal places. To prevent precision or rounding problems we take the following steps: We never explicitly calculate angles but rely on the three-point-orientation test, i.e. the position of a point c relative to the line through points a and b. Additionally, if points need to be placed somewhere on a polygon edge, an edge reference is stored to account for inaccuracies when calculating the point's coordinates.

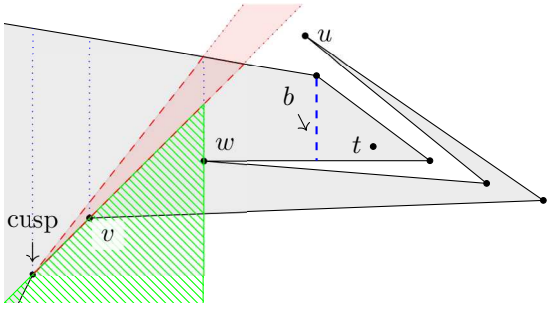


Figure 1: During the Jarvis march from the cusp to the diagonal b, the vertices need to be restricted to the shaded area. Otherwise, u would be considered part of the geodesic shortest path, as it is left of vw.

Delaunay and Trapezoid. In both algorithms, we need to find a piece of the shortest path as soon as the next diagonal is no longer visible from the current cusp. Asano, Mulzer, and Wang [3] only say that this should be done with a Jarvis march. During the implementation, we noticed that a naive Jarvis march with all vertices on P between the cusp and the next diagonal might include vertices that are not visible. Figure 1 shows an example: the vertex u would be included in the shortest path because it lies to the right of the cone and to the left of vw.

The solution for *Trapezoid* is to consider only vertices whose x-coordinate is between the cusp and the point where the visibility cone leaves P for the first time. For ease of implementation, one can also limit it to the x-coordinate of the last trapezoid boundary visible from the cusp. Figure 1 shows this region in

[1] `https://github.com/jonasc/constant-workspace-algos`

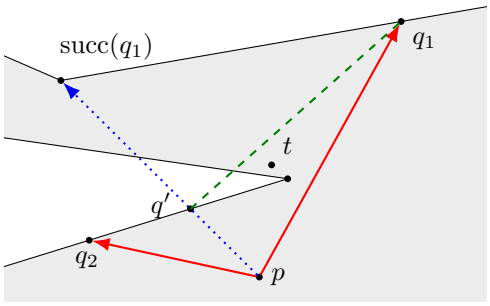


Figure 2: Asano et al. [2] state that one should check whether "t lies in the subpolygon from q' to q_1." Here, we should use q_1pq' to shrink the cutoff region.

green. For *Delaunay*, a similar approach can be used. The only difference is that the triangle boundaries are not all vertical lines.

Makestep. In our implementation of *Makestep*, we would like to point out an interesting detail; see Figure 2. The description by Asano et al. [2] says that to advance the visibiliy cone, we should check if "t lies in the subpolygon from q' to q_1." If so, the visibility cone should be shrunk to $q'pq_1$, otherwise to q_2pq'.

However, the "subpolygon from q' to q_1" is not clearly defined if the line segment $q'q_1$ is not contained in P. To avoid this difficulty, we consider pq' instead. This line segment is always in P, and it divides the cutoff region P' into two parts, a "subpolygon" between q' and q_1 and a "subpolygon" between q_2 and q'. Now we can easily choose the one containing t.

4 Experiments

Test set generation. Our experiments were conducted as follows: given a number of vertices n, we generate 4–10 random polygons, depending on n. For this, we use a tool developed in a software project at our department [7] which (among others) uses the *Space Partitioning* algorithm by Auer and Held [4].

For each edge e of each generated polygon, we find the incident triangle t_e of the constrained Delaunay triangulation. We add the barycenter of t_e to a point set S. Then, S has between $\lfloor n/2 \rfloor$ and $n-2$ points.

Test execution. For each pair of points from S, we find the shortest path using all of the implemented algorithms. Since the number of pairs grows quadratically, we restrict the tests to 1500 random pairs for all $n \geq 200$. We first run each algorithm once in order to determine the memory consumption. To obtain reproducible numbers, we disable the garbage collection functionality. After that, we run the algorithm between 5 and 20 times, depending on how long it takes. We measure the time for each run. We then take the median of the times as a representative for this point pair.

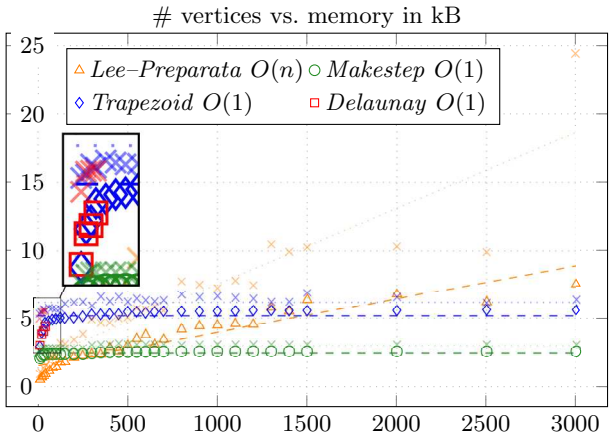


Figure 3: Memory consumption for random instances. The solid shapes are the median values; the transparent crosses are maximum values.

Test setup. Since we have a quadratic number of test cases, a lot of time is needed to run the tests. Thus the tests were distributed on multiple machines and on multiple cores. We had six computing machines, each with two quad-core CPUs. Three machines had Intel Xeon E5430 CPUs with 2.67 GHz; the other three had AMD Opteron 2376 CPUs with 2.3 GHz.

5 Results

The results of the experiments can be seen in the following plots. The plot in Figure 3 shows the median and maximum memory consumption as solid shapes and transparent crosses, respectively, for each algorithm and each input size. More precisely, the plot shows the median and the maximum over all polygons with a given size and over all pairs of points in each such polygon.

We observe that the memory consumption for *Trapezoid* and *Makestep* is always smaller than a certain constant. The shape of the median values might suggest logarithmic growth. However, a smaller number of vertices leads to a higher probability that s and t lie in the same triangle or can see each other. In this case, many geometric functions and subroutines, each of which requires an additional constant amount of memory, are not called. A large number of point pairs with only small memory consumption naturally entails a smaller median value.

The second plot in Figure 4 shows the median and the maximum running time in the same way as Figure 3. Not only does *Delaunay* have a cubic running time, but it also seems to have a quite large constant, as it grows much faster than the other algorithms.

In the lower part of Figure 4, we see the same x-domain, but with a much smaller y-domain. Here, we observe that *Trapezoid* and *Makestep* both have a quadratic running time; *Trapezoid* needs about two

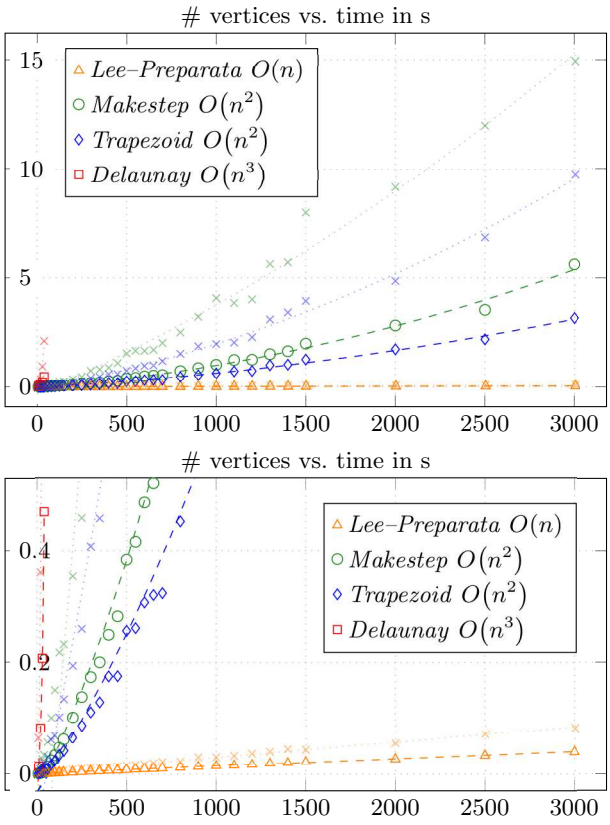


Figure 4: Runtime for random instances. Solid shapes are median values; transparent crosses are maximum values. The bottom plot is a scaled version of the top.

thirds of the time needed by *Makestep*. Finally, the linear-time behavior of *Lee–Preparata* is clearly visible.

We observed that the tests ran approximately 85 % slower on the AMD machines compared to the Intel servers. This reflects the difference between 2.3 GHz and 2.67 GHz. Since the tests were distributed equally on the machines it does not change the overall results.

6 Conclusion

We have implemented and experimented on three different constant-workspace algorithms for geodesic shortest paths in simple polygons. Not only did we observe the cubic worst-case running time of *Delaunay*, but we also noticed that the constant factor is rather large. This renders the algorithm useless already for polygons with a few hundred vertices, where the computation might, in the worst case, take several minutes.

As predicted by the theory, *Makestep* and *Trapezoid* exhibit the same asymptotic time and space consumption. *Trapezoid* has an advantage in the constant factor of the running time, while *Makestep* needs only about half as much memory. Since in both cases the memory requirement is bounded by a constant, *Trapezoid* would be our preferred algorithm.

We chose Python for the implementation mostly due to our experience, good debugging facilities, fast prototyping possibilities and the availability of numerous libraries. In hindsight, it might have been better to choose another programming language. Python's memory profiling and tracking abilities are limited, so that we cannot easily get a detailed view of the used memory with all the variables. Furthermore, a more detailed control of the memory management could be useful for performing more detailed experiments.

References

[1] T. Asano, K. Buchin, M. Buchin, M. Korman, W. Mulzer, G. Rote, and A. Schulz. "Memory-Constrained Algorithms for Simple Polygons". In: *CGTA* 46.8 (2013), pp. 959–969.

[2] T. Asano, W. Mulzer, G. Rote, and Y. Wang. "Constant-Work-Space Algorithms for Geometric Problems". In: *JoCG* 2.1 (2011), pp. 46–68.

[3] T. Asano, W. Mulzer, and Y. Wang. "Constant-Work-Space Algorithms for Shortest Paths in Trees and Simple Polygons". In: *JGAA* 15.5 (2011), pp. 569–586.

[4] T. Auer and M. Held. "Heuristics for the Generation of Random Polygons". In: *Proc. 8th Canada Conf. Comput. Geom.* Ottawa, 1996, pp. 38–43.

[5] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry*. Springer, 2008.

[6] L. P. Chew. "Constrained Delaunay Triangulations". In: *Algorithmica* 4 (1-4 1989), pp. 97–108.

[7] S. Dierker, M. Ehrhardt, J. Ihrig, M. Rohde, S. Thobe, and K. Tugan. *Abschlussbericht zum Softwareprojekt: Zufällige Polygone und kürzeste Wege*. Institut für Informatik, Freie Universität Berlin, Aug. 20, 2012. URL: `https://github.com/marehr/simple-polygon-generator`.

[8] S. Har-Peled. "Shortest Path in a Polygon Using Sublinear Space". In: *JoCG* 7.2 (2016), pp. 19–45.

[9] J. D. Hunter. "Matplotlib: A 2D Graphics Environment". In: *Computing In Science & Engineering* 9.3 (2007), pp. 90–95.

[10] D. T. Lee and F. P. Preparata. "Euclidean Shortest Paths in the Presence of Rectilinear Barriers". In: *Networks* 14.3 (Aut. 1984), pp. 393–410.

[11] Python Software Foundation. *Python*. Version 3.5. URL: `https://www.python.org/`.

[12] D. Rufat. *Python Triangle*. Version 20160203. 2016. URL: `http://dzhelil.info/triangle/` (visited on 12/05/2016).

[13] J. R. Shewchuk. "Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator". In: *Applied Computational Geometry towards Geometric Engineering*. Springer, 1996, pp. 203–222.

Computing Wave Impact in Self-Organised Mussel Beds

Johan van de Koppel* Maarten Löffler† Tim Ophelders‡

Figure 1: Mussels organised in strips.

Abstract

We model the effects of byssal connections made by mussels within patterned mussel beds on bed stability as a disk graph, and propose a formula for assessing which mussels, if any, would get dislodged from the bed under the impact of a wave. We formulate the computation as a flow problem, giving access to efficient algorithms to evaluate the formula. We then analyse the geometry of the graph, and show that we only need to compute a maximum flow in a restricted part of the graph, giving rise to a near-linear solution in practise.

1 Introduction

Mussel beds in the Waddenzee have attracted the interest of ecologists because they form typical self-organised patterns, consisting of strings of mussels that form reticulate networks, see Figure 1. Experimental studies have revealed that mussels form spatial patterns to provide stability against incoming waves, while still allowing enough access to food for individual mussels [3, 9]. To provide in-detail understanding of how the spatial structure of the mussel bed affects the persistence of individual mussels, the process of pattern formation has been modelled in individual-based models. However, these models simplify the impact of wave action on mussels, ignoring the protective effects of mussel clumps and strings on individual survival.

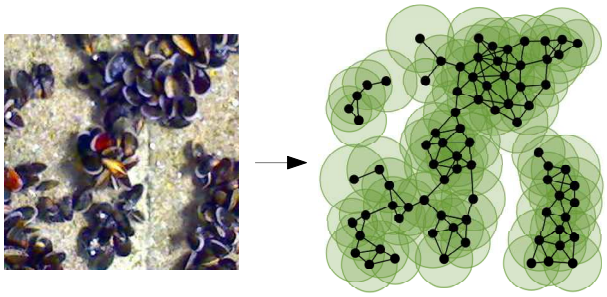


Figure 2: Mussels are modelled as a set of influence radius disks, and their relations define a disk graph.

In order to run simulations on a large enough scale for macrobiological effects to become visible, efficient algorithms to compute the stability of given mussel configuration are needed [5]. For this, a suitable model for mussel beds is needed, as well as efficient algorithms to compute the effect of waves. Models that include a large-scale group/structure effect on mussel survival may provide a better understanding of self-organisation in mussels, and stability of mussel beds as a key habitat to many species.

Based on their size, mussels connect themselves to anything solid within a given radius around them using byssal threads. On sand, the only solid objects are other mussels. As such, the graph of connected mussels is naturally modelled as a disk graph [8, 9]. Disk graphs have been studied extensively in computational geometry and discrete mathematics [1, 2, 4, 6].

We set out to leverage this mathematical and computational knowledge to understand how the topological network formed by mussel byssal threads influences mussels' survival/persistence. An important focus is to understand whether and how net-shaped structures provide a more stable landscape than loose clumps, where wave vulnerability is lower. When waves exert force on a limited section of the bed (say 25×25 cm), small clumps can easily get dislodged, while larger clumps that are connected to the larger bed, may persist in the bed. For this we need a binary test, that checks whether the mussels within the wave impact zone are sufficiently connected not to break free.

*Royal Netherlands Institute for Sea Research (NIOZ) and Utrecht University, Johan.van.de.Koppel@nioz.nl

†Dept. of Information and Computing Science, Utrecht University, m.loffler@uu.nl

‡Dept. of Mathematics and Computer Science, TU Eindhoven, t.a.e.ophelders@tue.nl

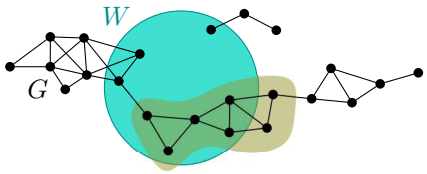


Figure 3: The wave impact zone W. In this case, the shaded set of mussels at the bottom might get dislodged: some mussels inside W stay because they are strongly connected to other mussels outside W, while some mussels outside W get pulled away along with mussels in W. In this case, two connections break.

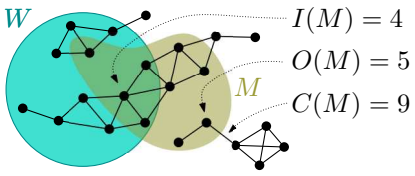


Figure 4: The potential of a given set M depends on $I(M)$, $O(M)$, and $C(M)$. In this particular case, the potential is most likely negative (depending on the weights), meaning this set of mussels would not get dislodged by W.

Contribution. We propose a model for approximating the impact of waves on mussel beds, and give a precise formulation of this model, including a formula for testing whether a given clump of mussels gets dislodged under influence of a given wave. We then show how to formulate the evaluation of this formula as a flow problem. Finally, we analyse the geometry of the underlying disk graph to speed up the computation of the maximum flow.

2 Modelling

We first present a mathematically precise formulation of the problem at hand.

2.1 Disk Graphs

Mussels grow connecting byssus threads to all other mussels within a given distance, depending on their age and size. Thus, we represent individual mussels by weighted points in $\mathbb{R}^2$. A set P of mussels then induces a graph G as follows. Each mussel (p_i, w_i) corresponds to a vertex at point p_i, and has edges to all mussels (p_j, w_j) for which $|p_i p_j| \leq w_i$. Note that the graph is undirected: once a byssus connection is established, it is no longer relevant which mussel created the connection.

Additionally, we know that mussels have a minimum and maximum size. The minimum size gives us a bound on the minimum distance δ^- between any two mussels, since mussels do not overlap; mussels vary in size from just under 1 cm in width up to 10 cm in length. The maximum size gives us a bound on the maximum distance δ^+ between two connected mussels, because threads do not grow longer than this distance; this length is not fully understood but it is not much longer than 10 cm. We define $\phi = \frac{\delta^+}{\delta^-}$ to be the ratio between this largest and smallest distance.

2.2 Wave Impact Zone

We are interested in what happens when a wave hits a part of the mussel bed. We model this by a disk W of radius r, which we call the *wave impact zone (WIZ)*. Figure 3 illustrates this.

We assume that when a wave hits a zone, all mussels in that zone are affected by the wave and start pulling on their neighbours to get washed away. Several things may happen:

- the mussels are washed away and pull their neighbours with them; or
- the neighbours keep the mussels anchored, and nobody is washed away; or
- some connections break, and some mussels are washed away while others stay behind.

2.3 Potential Function

Given G, W, and a set of mussels $M \subseteq P$, we define a function $F(M)$ that decides how much force is exerted on M by the wave. If $F(M)$ is positive, then there is enough force to dislodge M from G, breaking all connections between mussels in M and their neighbours outside M; if $F(M)$ is negative (or zero), then there is not enough force for this to happen. In order to define this $F(M)$, we are going to count three things:

The number of connections between M and the rest of the mussels:

$$C(M) = |\{(a, b) \in E : a \in M \wedge b \notin M\}|$$

The number of mussels of M that are inside the wave impact zone:

$$I(M) = |M \cap W|$$

The number of mussels of M that are outside the wave impact zone:

$$O(M) = |M \setminus W|$$

Figure 4 shows an example. The idea is that mussels in M that are in W are pulling on their neighbours, so they provide force. Mussels in M that are not in W need to be pulled, requiring force. Finally, connections that are broken require (significantly higher) force. This leads to a weighted formula, where w_C, w_I, and w_O are positive weights:

$$F(M) = w_I I(M) - w_O O(M) - w_C C(M)$$

2.4 Objective

To find out whether any mussels get dislodged, we take the maximum of $F(M)$ over all sets M, and see if this is a positive number. We also want to know *which* mussels get dislodged. If multiple sets of mussels have a positive potential, then it is not clear a priori which of the sets gets dislodged by the wave.

Lemma 1 *If we remove the set M that maximises F from G, then all remaining sets have negative or zero potential in the resulting graph.*

To prove the lemma, we first show some intermediate properties of the potential function.

Observation 1 *For sets of mussels A and B, we have $F(A \cap B) + F(A \cup B) = F(A) + F(B) + 2w_C E(A, B)$, where $E(A, B)$ counts the number of edges that have exactly one endpoint in A and exactly one endpoint in B.*

Proof. We verify the three components of F separately. The I term counts mussels inside W inside the sets. By definition, $W \cap A$ and $W \cap B$ overlap in $W \cap A \cap B$, so the sum of their cardinalities is the cardinality of the union plus the cardinality of the intersection. The same argument holds for the O term. Finally, for the C term, we observe that on both sides of the equation we count exactly those edges that cross the boundary of A or the boundary of B. □

This observation implies that whenever we have two candidate sets, either their intersection or their union has a higher potential than the lowest potential of the two. This allows us to prove Lemma 1.

Corollary 1 $F(A \cap B) \uparrow F(A \cup B) \geq F(A) \downarrow F(B)$.

Proof. [of Lemma 1] Assume for contradiction that M is the set of maximum potential, and that after removing M from G, another set N now has positive potential as well. Now consider the set $M \cup N$. By assumption, $F(M \cup N) \leq F(M)$. However, M and N are disjoint, so $F(M \cup N) = F(M) + F(N) + 2w_C E(M, N)$. This means $F(N) + 2w_C E(M, N)$ must be negative or zero. However, after removing M from G, the potential of N increases by only $w_C E(M, N)$, after which it should, by assumption, become positive. This is clearly a contradiction. □

Lemma 1 implies that the model is well-formed: if there are multiple sets of mussels with positive potential, we simply select the one with the maximum potential. Further note that if there are multiple mussel sets with maximum potential, then one of them contains all others; this is the one we wish to report.

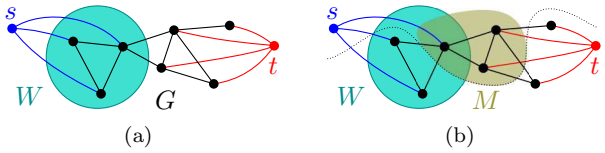


Figure 5: (a) The augmented graph G'. A new node s is connected to all nodes inside W, and a new node t is connected to all nodes outside W. Black edges have weight w_C, blue edges have weight w_I, and red edges have weight w_O. (b) The cut (dotted) associated with a given set M.

3 Algorithms

3.1 Constructing G

We first investigate how to construct the graph G from a given set of weighted points. We can easily construct the graph in $O(n^2)$ time, by testing, for each pair (p_i, w_i) and (p_j, w_j), whether $|p_i p_j| \leq \max(w_i, w_j)$.

However, we note that in realistic cases (such as cases where ϕ is bounded), the number of edges will be significantly smaller than n^2, and we can compute the graph more efficiently by first storing the points in a suitable data structure, and only testing pairs of points that are sufficiently close to each other.

We first compute a range tree with fractional cascading on P in $O(n \log n)$ time. We then search locally, for each point p_i, for all other points at distance at most w_i from p_i. Since $w_i \leq \delta^+$ and the minimum distance between points is δ^-, there are at most ϕ^2 such points. For each of these points, we add an edge.

Theorem 2 *Given a set P of n mussels in the plane, we can construct G in $O(n(\log n + \phi^2))$ time.*

Given additionally the wave impact zone W, we can easily augment G in linear time (see next section).

3.2 Min Cut Formulation

We can compute minimum value of F using a max flow algorithm, as follows.

We weight the original edges of G by w_C. We augment the graph G with two artificial nodes s and t. We add an edge from s to every node in W, weighted by w_I, and an edge from every node outside W to t, weighted by w_O. We call the resulting augmented weighted graph G'. Figure 5(a) shows an example.

Define $L = w_I |W \cap P|$ to be the maximum amount of force a given wave can exert on the mussels. The following lemma relates the value of a min cut in G' to $F(M)$ and L; see also Figure 5(b).

Lemma 3 *A minimum cut in G' that separates s from t corresponds with a set M that maximises $F(M)$. The value of $F(M)$ is L minus the weight of the minimum cut.*

Proof. Let M be any set of vertices of G. The cut in G' that separates $M \cup \{s\}$ from $P \setminus M \cup \{t\}$ has a total weight of $w_C C(M) + w_I|W \cap P \setminus M| + w_O|M \setminus W|$, which is equal to $L - F(M)$. Clearly, the set M that minimises the weight of the cut, maximises F. □

Max flow can be solved in $O(|V| \cdot |E|)$ time or slightly faster (for instance, see [7]); we compute a minimum cut in G' in $O(|P| \cdot |E|)$ time, which gives us a $O(n^3)$ algorithm to solve the problem on arbitrary graphs. Assuming a bound on the mussel density ϕ, we can immediately improve this by observing that the maximum number of edges is only $O(\phi^2 n)$ instead of $O(n^2)$.

Theorem 4 *Given a set P of n mussels in $\mathbb{R}^2$, and a wave impact zone W, we can compute the set of mussels M that maximises $F(M)$ in $O(\phi^2 n^2)$ time.*

3.3 Geometric Analysis

By exploiting the geometry of unit disk graphs, and making reasonable assumptions about the wave impact zone and the weights w_I, w_O, and w_C, we can improve further on the running time of the above algorithm. The basic observation is that, in order to compute a maximum flow in G', we need never use any mussels that are too far from the boundary of W.

We assume that w_C, w_I, and w_O are within a constant factor of each other. We also assume that the wave impact zone W is a circle of radius r, where r is constant and $r > \delta^+$. For ease of presentation we assume W.L.O.G. that $\delta^- = 1$.

We claim that, to compute the min cut in G', we can restrict our attention to the graph G'' composed of those mussels that are at distance at most $\frac{w_C}{w_O} r\phi^3$ from W. There are at most $\frac{w_C}{w_O} r^2\phi^4$ nodes in G''.

Lemma 5 *The min cut in G' is the same as the min cut in G''.*

Proof. Let F be the set of edges of G that cross the boundary of W. First, we observe that any flow from s to t must use at least one edge of F. All mussels that contribute to edges in F must lie in an annulus centered at the boundary of W of width 2ϕ. Since the radius of W is r, there can be at most $r\phi$ mussels in this annulus, and each can have an edge to at most ϕ^2 other mussels. So, we have $|F| \leq r\phi^3$. Each edge has capacity w_C, so the maximum value the flow problem can have is bounded by $\mathcal{Q} = r\phi^3 w_C$.

Now, we argue that it never makes sense to use any mussels farther than $\mathcal{Q}\frac{\phi}{w_O}$ from W outside W, or farther than $\mathcal{Q}\frac{\phi}{w_I}$ from W inside W. Indeed, any path that long encounters enough edges to s or t to accommodate all the flow. □

The number of mussels in this area is $\mathcal{Q}\frac{\phi}{w_O} r = \frac{w_C}{w_O} r^2\phi^4$.

We can find those mussels easily once G is known. Therefore, we can substitute n in the result of the previous section, and arrive at the following result.

Theorem 6 *Given a set P of n mussels in $\mathbb{R}^2$, and a wave impact zone W, we can compute the set of mussels M that maximises $F(M)$ in $O(n \log n + n\phi^2 + (\frac{w_C}{w_O})^2 r^4 \phi^{10})$ time.*

4 Future Work

We modelled and analysed the effect of a single wave on a mussel bed, and gave efficient algorithms to compute it. The next step is to investigate whether we can calculate the value of F for multiple wave impact zones more efficiently, since a large number of waves need to be evaluated in a single simulation.

Acknowledgments M.L. and T.O. are supported by the Netherlands Organisation for Scientific Research (NWO) under project no. 639.021.123 and 614.001.504 (M.L.) and 639.023.208. (T.O.).

References

[1] J. L. Bentley, D. F. Stanat, and J. E. Hollins Williams. The complexity of finding fixed-radius near neighbors. *Information Processing Letters*, 6(6):209–212, 1977.

[2] H. Breu. *Algorithmic Aspects of Constrained Unit Disk Graphs*. PhD thesis, 1996.

[3] M. de Jager, F. J. Weissing, P. M. J. Herman, B. A. Nolet, and J. van de Koppel. Levy walks evolve through interaction between movement and environmental complexity. *Science*, 332(6037):1551–1553, 2011.

[4] W. K. Hale. Frequency assignment: Theory and applications. *Proceedings of the IEEE*, 68(12):1497–1514, 1980.

[5] Q.-X. Liu, P. M. J. Herman, W. M. Mooij, J. Huisman, M. Scheffer, H. Olff, and J. van de Koppel. Pattern formation at multiple spatial scales drives the resilience of mussel bed ecosystems. *Nature Communications*, 5(5234), 2014.

[6] M. Marathe, H. Breu, H. H. II, S. Ravi, and D. Rosenkrantz. Simple heuristics for unit disk graphs. *Networks*, 25(59):68, 1995.

[7] J. B. Orlin. Max flows in $o(nm)$ time, or better. In *STOC '13 Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 765–774, 2013.

[8] J. Sherratt and J. Mackenzie. How does tidal flow affect pattern formation in mussel beds? *J. Theor. Biol*, 406:83–92, 2016.

[9] J. van de Koppel, J. C. Gascoigne, G. Theraulaz, M. Rietkerk, W. M. Mooij, and P. M. J. Herman. Experimental evidence for spatial self-organization and its emergent effects in mussel bed ecosystems. *Science*, 322(5902):739–742, 2008.

A Novel MIP-based Airspace Sectorization for TMAs*

Tobias Andersson Granberg, Tatiana Polishchuk, Christiane Schmidt†

Abstract

We present a MIP-based airspace sectorization framework for Terminal Maneuvring Areas incorporating an airspace complexity representation; it easily deals with convex sectors. It is also the first step towards an integrated design of routes and sectorization.

1 Introduction

Over the last decades air traffic volumes have increased, and projections indicate that the growth will continue. The resulting congestion is particularly concentrated on Terminal Maneuvring Areas (TMAs), i.e., the area surrounding one or more neighboring aerodromes, as traffic converges towards a point near the runway. An optimized design of the TMA control sectors can increase capacity (the sectors partition the airspace above the TMA territory). The human factor is a major challenge for this design: each sector is monitored by an air traffic controller (ATCO). The mental workload associated with working in such a complex system leads to the major constraints for sectorization: the workload should be balanced and below thresholds for every single ATCO.

Today, the airspace layout at most airports is done manually. Many authors suggested automatic design methods for sectorization, for an extensive survey see Flener and Pearson [3], but the vast majority of approaches concentrate on en-route airspace.

Taskload/Workload. ATCOs must first of all ensure safe separation of aircraft (i.e., ensure a minimum safety distance between aircraft). In addition, they enable aircraft to reach their destinations in a timely manner. To do so, they permanently anticipate and detect (potential) conflicts and perform various other tasks that contribute to the airspace's complexity and drive an ATCO's mental workload. Both taskload and workload reflect the demand of the air traffic controller's monitoring task (they measure objective demand and subjective demand experienced during a task, resp.); we will refer to both as taskload. Loft et al. [4] survey methods for the elaborate problem of determining the taskload of a sector.

*This research is funded by grant 2014-03476 (ODESTA: Optimal Design of Terminal Airspace) from Sweden's innovation agency VINNOVA and in-kind participation of LFV. We thank Billy Josefsson (project co-PI, LFV) and the project reference group for discussions of sectorization design.

†ITN, Linköping University, Sweden, {tobias.andersson.granberg, tatiana.polishchuk, christiane.schmidt}@liu.se

Recently, Zohrevandi et al. [5] presented a model for relating ATCO's taskload to the airspace complexity. They quantify the taskload as a weighted combination of ATCOs' clicks on the radar screen (weight≈time for the task). Using linear regression the authors were able to explain terminal airspace complexity, given by eight complexity factors, about 70% better than the model by Djokic et al. [1] (who used controller pilot (C-P) data link communication and C-P voice communications). Thus, the weighted radar screen clicks is a very good model for terminal airspace complexity. We use the heat maps for weighted clicks([5]) as input. Our model does not depend on these specific maps, it is a general model that integrates complexity.

Notation and Preliminaries. A *sectorization* of a simple polygon P is a partition of P into k disjoint subpolygons $S_1 \dots S_k$ ($S_i \cap S_j = \emptyset \ \forall i \neq j$), such that $\cup_{i=1}^{k} S_i = P$. The subpolygons S_i are called *sectors*.

Sectorization Problem:
Given: The coordinates of the TMA, defining a polygon P, the number of sectors $|\mathcal{S}|$, and a set $\mathcal{C}$ of constraints on the resulting sectors.
Find: A sectorization of P with $k = |\mathcal{S}|$, fulfilling $\mathcal{C}$.

2 Grid-based MIP formulation

We discretize the search space by laying out a square grid in the TMA. Every grid node has directed edges to its 8 neighbors ($N(i)$ = set of neighbors of i (including i)), resulting in a bidirected graph $G = (V, E)$. The length of an edge $(i, j) \in E$ is denoted by $\ell_{i,j}$.

The main idea for the sectors is to use an artificial sector, S_0, that encompasses the complete boundary of P, using all counterclockwise (ccw) edges. That is, we use sectors in $\mathcal{S}^* = \mathcal{S} \cup S_0$ with $\mathcal{S} = \{S_1 \dots S_k\}$. For all edges (i, j) used for boundary of any sector, we enforce that also the opposite edge, (j, i), is used for another sector, see Fig. 1(a). We use decision variables $y_{i,j,s}$, where $y_{i,j,s} = 1$ indicates that edge (i, j) is a boundary edge for sector s. We add:

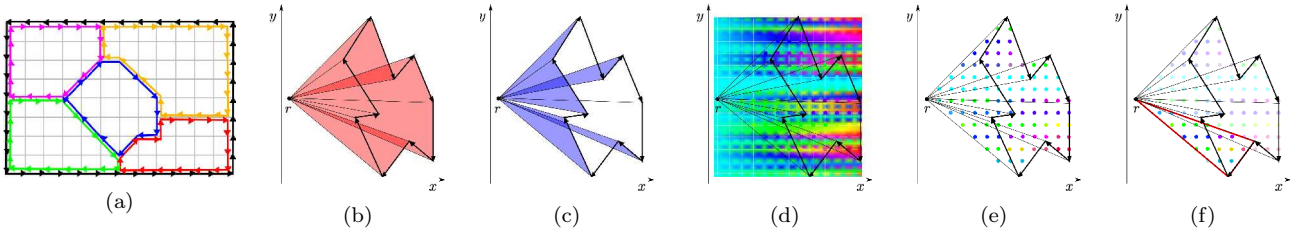
$$y_{i,j,0} = 1 \qquad \forall (i,j) \in S_0 \tag{1}$$

$$\sum_{s \in \mathcal{S}^*} y_{i,j,s} - \sum_{s \in \mathcal{S}^*} y_{j,i,s} = 0 \qquad \forall (i,j) \in E \tag{2}$$

$$y_{i,j,s} + y_{j,i,s} \leq 1 \quad \forall (i,j) \in E, \forall s \in \mathcal{S}^* \tag{3}$$

$$\sum_{s \in \mathcal{S}^*} y_{i,j,s} \leq 1 \qquad \forall (i,j) \in E \tag{4}$$

$$\sum_{(i,j) \in E} y_{i,j,s} \geq 3 \qquad \forall s \in \mathcal{S}^* \tag{5}$$



(a) (b) (c) (d) (e) (f)

Figure 1: (a) Artificial sector S_0 (black) and a sectorization with $|\mathcal{S}| = 5$. Edges are slightly offset to enhance visibility. (b)/(c) Area of polygon P (bold): each edge of P forms an oriented triangle with a reference point r. Cw triangles contribute positive (b), ccw triangles negative (c). (d)-(f) Heat value extraction for a triangle.

$$y_{i,j,s} \in \{0,1\} \quad \forall (i,j) \in E, \forall s \in \mathcal{S}^* \tag{6}$$

Eq. (1) ensures that all ccw boundary edges belong to S_0. Consistency between edges is given by Eq. (2): if (i,j) is used for some sector, edge (j,i) has to be used as well. Eq. (3) ensures that a sector cannot contain both edges (i,j) and (j,i). With Eq. (2) it ensures that if an edge (i,j) is used for sector S_ℓ, the edge (j,i) has to be used by some sector $S_k \neq S_\ell$. Eq. (4) enforces that one edge cannot participate in two sectors. Eq. (5) enforces a minimum size for all sectors. We add constraints on the degree of vertices:

$$\sum_{l \in V:(l,i) \in E} y_{l,i,s} - \sum_{j \in V:(i,j) \in E} y_{i,j,s} = 0 \quad \forall i \in V, \forall s \in \mathcal{S}^* \tag{7}$$

$$\sum_{l \in V:(l,i) \in E} y_{l,i,s} \leq 1 \quad \forall i \in V, \forall s \in \mathcal{S}^* \tag{8}$$

Eq. (7) yields indegree=outdegree for all vertices. By Eq. (8) a node has at most one ingoing edge per sector.

Constraints (1)-(8) guarantee that the union of the $|\mathcal{S}|$ pairwise disjoint sectors completely covers the TMA. Of course, there are various other constraints for a sectorization, see [3]. The constraints we consider can roughly be split in two categories: geometric and balancing. **Balancing Constraints** are related to two factors: size/area and taskload. We consider:

- 𝔞) *Balanced size*: The area of each sector must be balanced out with the area of other sectors.
- 𝔟) *Bounded taskload*: There is an upper bound of movements that an ATCO can handle per hour.
- 𝔠) *Balanced taskload*: The taskload of each sector, and, thus, of each ATCO, must be balanced out with the taskload of other sectors.

For 𝔞, balanced size, we need to assign an area to the sector selected by the boundary edges. The area of a polygon P with rational vertices is rational, and can be computed efficiently (see Fekete et al. [2]): we introduce a reference point r, and compute the area of the triangle of each directed edge e of P and r, see Fig. 1(b)/(c). We then sum up the triangle area for all edges of P: cw and ccw triangles contribute positive and negative, respectively. Let $f_{i,j}$ denote the signed area of the triangle formed by (i,j) and r.

$$\sum_{(i,j) \in E} f_{i,j}\, y_{i,j,s} - a_s = 0 \quad \forall s \in \mathcal{S}^* \tag{9}$$

$$\sum_{s \in \mathcal{S}} a_s = a_0 \tag{10}$$

Eq. (9) assigns the area of sector s to the variable a_s, Eq. (10) ensures that the sum of the a_s's equals the area of the complete TMA. If we want to balance the sector size, we add $a_s \geq a_{LB} \quad \forall s \in \mathcal{S}$ (11) to the IP. It gives a lower bound on the size of each sector, we use $a_{LB} = c_1 \cdot a_0/|\mathcal{S}|$, with , e.g., $c_1 = 0.9$.

For constraints 𝔟 and 𝔠 we need to be able to associate a taskload with a sector. Here, we assume that a heatmap representing the controller's taskload is given, see Section 1. Given this heatmap we overlay it with a grid, extract the value at the grid points, see Fig. 1(d), and use this discretized heatmap, see Fig. 1(e), for further computations. We associate each discrete heatmap point, q, with a "heat value", h_q. Again, we consider triangles for each directed edge (i,j) of P and the reference point r, see, e.g, Fig. 1(f): we sum up the heat values for all grid points within the triangle. The sign of the heat value for a triangle is determined by the sign of $f_{i,j}$, denoted by $p_{i,j}$. Let $h_{i,j}$ denote the signed heat value of the triangle formed by edge (i,j) and r: $h_{i,j} = p_{i,j} \sum_{q \in \Delta(i,j,r)} h_q$. If the taskload is of interest, we add Eq. (12), which assigns each sector s a taskload t_s. In analogy to the balanced size, we add Eq. (13) to achieve a balanced taskload. Here, $t_{LB} = c_2 \cdot t_0/|\mathcal{S}|$ with, e.g., $c_2 = 0.9$. For a bounded taskload we add Eq. (14), with some fixed upper bound t_{UB} on the taskload in any sector.

$$\sum_{(i,j) \in E} h_{i,j}\, y_{i,j,s} - t_s = 0 \quad \forall s \in \mathcal{S} \tag{12}$$

$$t_s \geq t_{LB} \quad \forall s \in \mathcal{S} \tag{13}$$

$$t_s \leq t_{UB} \quad \forall s \in \mathcal{S} \tag{14}$$

Geometric Constraints:

- 𝔡) *Connected sectors*: A sector must be a connected portion of airspace, see Flener and Pearson [3].
- 𝔢) *Nice shape*: A sector should have a smooth boundary and an easily memorable shape, see [3].
- 𝔣) *Convex sectors*: The sectors should be convex: convexity can be defined either geometrically or trajectory-based, i.e., no route enters the same sector more than once, see Flener and Pearson [3].

$\mathfrak{g}$) *Interior conflict points:* Points that require increased attention from ATCOs should lie in the sector's interior.

For $\mathfrak{d}$ we chose to use the length of the sector boundary as an objective function. For constraint $\mathfrak{g}$ we cannot use an absolute threshold heat value for points on the sector boundary: we like to enforce points of relatively high airspace complexity, represented by heat values, to be in the interior. Again, we use the objective function. We take care of $\mathfrak{e}$ in postprocessing: Given constraint set $\mathcal{C}, \mathfrak{e} \in \mathcal{C}$, we solve the IP with $\mathcal{C} \setminus \{\mathfrak{e}\}$ and then use shortcuts by removing vertices as long as the constraints in $\mathcal{C} \setminus \{\mathfrak{e}\}$ are not violated.

We can easily integrate convexity in our approach—a feature many other optimization approaches lack. For a convex sector there exist only one connected chain of edges with cw triangles, and one connected chain of edges with ccw triangles, see Fig. 2(a). Unfortunately, the only-if-part of that statement is not true, see Fig. 2(b). We can make use of the fact that we have only eight edge directions. For every direction of an incoming edge, there are three directions of outgoing edges that are forbidden in a convex polygon, see Fig. 2(c): there exist two open cones (indicated in gray) in which a reference point must be located to detect the switch. Thus, any reference point located in the dark gray cones yields a switch in the triangle orientation. We place one reference point in each of the eight cones in Fig. 2(d), and denote them by $r_1, \ldots, r_8$ ($r = r_m$, for some $m \in \mathcal{M} = \{1, \ldots, 8\}$). At least one of the r_m will result in a cw/ccw switch for non-convex polygons. Let $p_{i,j,m}$ denote the sign of the triangle of the edge (i, j) and reference point r_m, $m \in \mathcal{M}$. We add:

$$q^s_{j,m} = \frac{1}{2}\left(\sum_{i:(i,j)\in E} p_{i,j,m}\, y_{i,j,s} - \sum_{l:(j,l)\in E} p_{j,l,m}\, y_{j,l,s}\right) \quad (15)$$
$$\forall s \in \mathcal{S},\ \forall j \in V,\ \forall m \in \mathcal{M}$$

This constraint assigns, for each sector, a value of -1,0,1 to each vertex. An interior vertex of either a chain of cw or ccw triangles has $q^s_{j,m} = 0$; if at j a chain with ccw (cw) triangles switches to a chain of cw (ccw) triangles $q^s_{j,m} = -1$ ($q^s_{j,m} = 1$). For a convex sector s, the sum over the $|q^s_{j,m}|$ for all sector vertices j is 2 for all reference points r_m; for non-convex sectors this value is larger than 2 for at least one reference point r_m. Eq.s (16),(17) define the absolute values. To enforce convexity (18) must hold. To this end we use Eq.s (19)-(22)to define variables $z^s_{i,j,m} = y_{i,j,s} \cdot qabs^s_{j,m} \forall i, j \in V \forall s \in \mathcal{S},\ \forall m \in \mathcal{M}$, and add Eq. (23).

$$qabs^s_{j,m} \geq\ \ q^s_{j,m}\ \forall s \in \mathcal{S}, \forall j \in V,\ \forall m \in \mathcal{M} \quad (16)$$
$$qabs^s_{j,m} \geq -q^s_{j,m}\ \forall s \in \mathcal{S}, \forall j \in V,\ \forall m \in \mathcal{M} \quad (17)$$
$$\sum_{i\in V}\sum_{j\in V} y_{i,j,s} \cdot qabs^s_{j,m} = 2\ \ \forall s \in \mathcal{S},\ \forall m \in \mathcal{M} \quad (18)$$
$$0 \leq\ \ z^s_{i,j,m} \forall i, j \in V\ \forall s \in \mathcal{S},\ \forall m \in \mathcal{M} \quad (19)$$
$$z^s_{i,j,m} \leq qabs^s_{j,m} \forall i, j \in V\ \forall s \in \mathcal{S},\ \forall m \in \mathcal{M} \quad (20)$$
$$z^s_{i,j,m} \leq\ \ y_{i,j,s} \forall i, j \in V\ \forall s \in \mathcal{S},\ \forall m \in \mathcal{M} \quad (21)$$
$$z^s_{i,j,m} \geq y_{i,j,s} - 1 + qabs^s_{j,m} \forall i, j \in V\ \forall s \in \mathcal{S},\ \forall m \in \mathcal{M} \quad (22)$$
$$\sum_{i\in V}\sum_{j\in V} z^s_{i,j,m} = 2\ \forall s \in \mathcal{S},\ \forall m \in \mathcal{M} \quad (23)$$

If we allow usage of a few reflex vertices, we might penalize reflex vertices in the objective function.

Objective Function. As opposed to most LP approaches, in our case, it is not obvious what kind of objective function should be used. Cost functions used in literature, cp. [3], are, e.g., taskload imbalance (constraint $\mathfrak{c}$), and number of sectors (which we consider as input). Our basic objective function is:

$$\min \sum_{s\in\mathcal{S}} \sum_{(i,j)\in E} \ell_{i,j} y_{i,j,s} \quad (24)$$

If for the balancing constraints we have $\mathfrak{a} \in \mathcal{C}, \mathfrak{b}, \mathfrak{c} \notin \mathcal{C}$, that is, we want to balance the area of the sectors, but are not interested in the sector taskload, objective function (24) ensures that sectors are connected, that is, we take care of constraint $\mathfrak{d}$, see Fig. 2 (e).

If we consider taskload, objective function (24) only yields connected sectors if c_2 in $t_{LB} = c_2 \cdot t_0/|\mathcal{S} \setminus S_0|$ of constraint (13) allows it: for example $c_2 = 0.9$ may not allow a "c_2-balanced" sectorization with connected sectors, but if we allow for larger disparities between sectors, making a connected solution feasible by lowering the parameter, e.g., $c_2 = 0.7$, we again obtain connected sectors. Essentially, this translates to: given the current complexity map a user must allow larger imbalances between controller's taskload, if having connected sectors is a necessary condition.

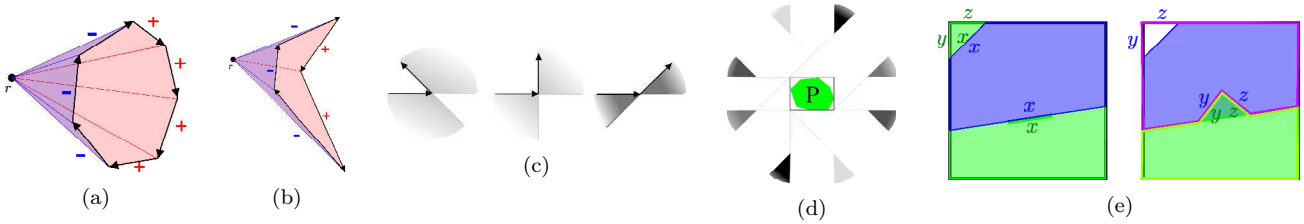
Integration of Constraint $\mathfrak{g}$. If $\mathfrak{g} \in \mathcal{C}$ we use:

$$\min \sum_{s\in\mathcal{S}} \sum_{(i,j)\in E} (\gamma\ell_{i,j} + (1-\gamma)w_{i,j})\, y_{i,j,s},\ \ 0 \leq \gamma < 1 \quad (25)$$

Where $w_{i,j}$ represents an edge weight that depends on the heat-values of its endpoints. We choose (I) $w_{i,j} = h_i + h_j$. or (II) $w_{i,j} = \sum_{k\in N(i)} h_k + \sum_{l\in N(j)} h_l$. (I) ensures that relatively large heat-values are not located on the sector boundary, (II) pushes larger values further into the interior. An alternative is to use a constraint with an upper bound W. This shows that we obtain an optimal connected solution, if, given c_2 and W, there exists a feasible connected solution.

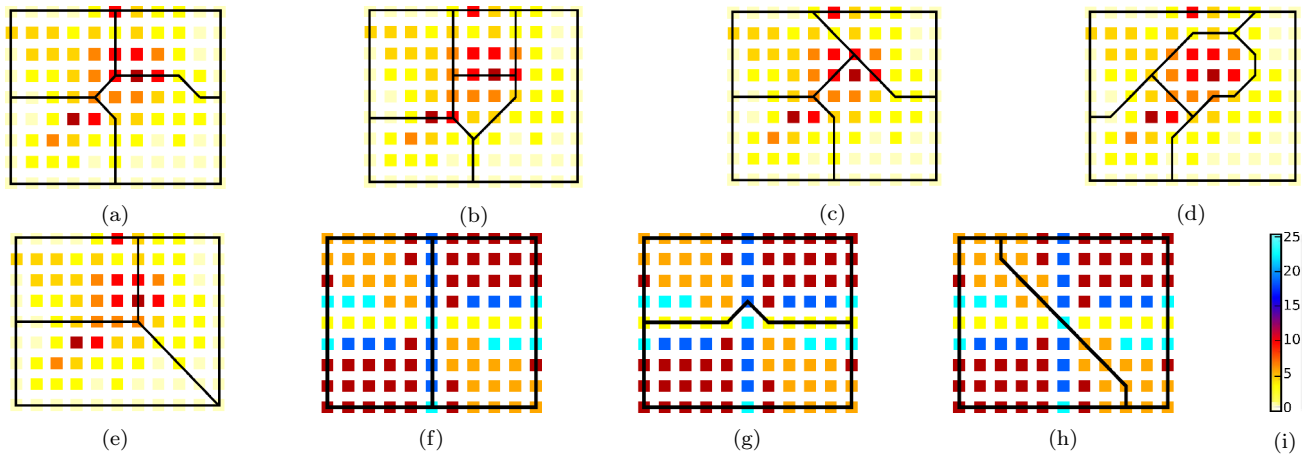
3 Experimental Study: Arlanda Airport

The model was solved using AMPL and CPLEX 12.6 on a single server with 24GB RAM and four kernels running on Linux. Each instance was run until a solution with less than 1% gap had not been found, or



(a) (b) (c) (d) (e)

Figure 2: A convex (a) and a non-convex (b) polygon. (c) Three outgoing edge directions yield a non-convex polygon (interior of P below ingoing edge). (d) Eight cones for detecting non-convexity. (e) Disconnected sectors are not optimal for (24). The sectors must completely cover the TMA. Assume there is a disconnected sector, like the green sector in the left, we can merge and decrease the total perimeter, we have: $(y+z+2x)+(2x) \leq (y+z)+(2y+2z)$ by triangle inequality.



(a) (b) (c) (d) (e) (f) (g) (h) (i)

Figure 3: (a), (b), (f) $\mathcal{C}_1 = \{\mathfrak{c}, \mathfrak{d}, \mathfrak{e}\}$. (a) $|\mathcal{S}| = 4$, and (b) $|\mathcal{S}| = 5$. (c), (d), (g), (h) $\mathcal{C}_2 = \{\mathfrak{c}, \mathfrak{d}, \mathfrak{e}, \mathfrak{g}\}$. (e) $\mathcal{C}_3 = \{\mathfrak{c}, \mathfrak{d}, \mathfrak{e}, \mathfrak{f}, \mathfrak{g}\}$, $|\mathcal{S}| = 3$. (c), (d) $|\mathcal{S}| = 4$, and (f)-(h) $|\mathcal{S}| = 2$. (c),(d),(e),(g) with $w_{i,j} = h_i + h_j$; (c): $\gamma = 0.9$, (d), (e), (g): $\gamma = 0.5$. (h) with $w_{i,j} = \sum_{k \in N(i)} h_k + \sum_{l \in N(j)} h_l$, $\gamma = 0.5$. (i) Color scale for heat values.

for a maximum of one CPU-hour. No instance finished with an optimality gap of more than 6%. If not mentioned otherwise we use $c_2 = 0.9$. For Stockholm TMA we do not think that convex sectors are a major concern: integrating the constraints in the IP is costly, while in the easy structure of the TMA a few reflex vertices will not result in a problem.

Fig. 3 (a)/(b) depicts solutions for $\mathcal{C}_1 = \{\mathfrak{c}, \mathfrak{d}, \mathfrak{e}\}$. Fig. 3(c)/(d) shows sectorizations for $|\mathcal{S}| = 4$, and $\mathcal{C}_2 = \{\mathfrak{c}, \mathfrak{d}, \mathfrak{e}, \mathfrak{g}\}$ for different values of γ. Comparing Fig. 3(c) with Fig. 3(a) we can observe that the objective to have interior conflict points avoids the heat value of 10 (dark red) in the center; for $\gamma = 0.5$, Fig. 3(d), both hotspots, i.e., areas of high heat values, are avoided by sector boundaries. Fig. 3(e) shows a convex sectorization for $|\mathcal{S}| = 3$, $\mathcal{C}_3 = \{\mathfrak{c}, \mathfrak{d}, \mathfrak{e}, \mathfrak{f}, \mathfrak{g}\}$ and weight function (I) with $\gamma = 0.5$.

Influence of choosing $w_{i,j}$. We present an instance (not connected to Stockholm TMA) to highlight the influence of the weight $w_{i,j}$. To ease perception we use $|\mathcal{S}| = 2$: we pick one cut through the rectangle. We consider $\mathcal{C}_2 = \{\mathfrak{c}, \mathfrak{d}, \mathfrak{e}, \mathfrak{f}\}$. In Fig. 3 (f) we use $\gamma = 1$, i.e., hotspots are neglected. Consequently, the cut runs along the shortest connection that balances workload. In Fig. 3 (g) and (h) we use $\gamma = 0.5$ and weight function (I) and (II), respectively. That is, in (g) we want to avoid that the sector boundary runs through hotspots; in the solution, we see that the low value heat points (yellow) are chosen. In (h), we also account for the weight of the neighbors of vertices on the cut. Thus, the cut that was optimal for (I) has a high weight. The optimal solution now runs through the areas of low complexity and avoids the hotspots.

References

[1] J. Djokic, B. Lorenz, and H. Fricke. Air traffic control complexity as workload driver. *Transp. Res. Part C: Emerging Technologies*, 18(6):930 – 936, 2010.

[2] S. P. Fekete, S. Friedrichs, M. Hemmer, M. Papenberg, A. Schmidt, and J. Troegel. Area- and boundary-optimal polygonalization of planar point sets. In *EuroCG 2015*, pages 133–136, 2015.

[3] P. Flener and J. Pearson. Automatic airspace sectorisation: A survey. *CoRR*, abs/1311.0653, 2013.

[4] S. Loft, P. Sanderson, A. Neal, and M. Mooij. Modeling and predicting mental workload in en route air traffic control: Critical review and broader implications. *Human Factors*, 49(3):376–399, 2007.

[5] E. Zohrevandi, V. Polishchuk, J. Lundberg, Å. Svensson, J. Johansson, and B. Josefsson. Modeling and analysis of controller's taskload in different predictability conditions. In *6th SESAR Innovation Days*, 2016.

A Combinatorial Upper Bound on the Length of Twang Cascades *

Leon Sering

Abstract

Damian et al. [2] introduced an intuitive type of transformations between simple polygons on a finite set of points in the plane. Each transformation consists of a sequence of atomic modifications of two types, called *stretches* and *twangs*. They proved that, for a given set of n points, the space of these simple polygons is connected by $O(n^2)$ such transformations.

We solve an open question of Damian et al. concerning a combinatorial upper bound on the length of these sequences. To this end, we show that the length of a twang cascade is bounded by $n^3/2$.

1 Introduction

This paper studies simple polygons on a fixed set S of n points in the plane. A *simple polygon* on S is a crossing-free cycle of straight line segments, each connecting two points of S such that every point of S is visited exactly once. In the following we abbreviate "simple polygon" by "polygon".

Sampling Polygons We consider the following challenging open problem:

Given a point set S as input, the task is to generate a random polygon on S with *uniform distribution*, that is, if r is the number of polygons on S, we want to choose one with a probability of $\frac{1}{r}$. So far no algorithm is known to do this efficiently.

However, there are various partial solutions for this problem. On the one hand, there are efficient generators for specific subsets of simple polygons such as x-monotone polygons [5] or star-shaped polygons [1]. On the other hand, there are algorithms that produce all possible polygons with positive probability but they are not generated uniformly at random. For example, the *2-Opt-Moves* algorithm [1] produces a random permutation of S and then removes all self-intersections step by step.

Random Walk Approach A different approach is to repeatedly apply random modifications to a polygon, a so-called *Markov chain Monte Carlo sampling*. We define a class of transformations, each turning one polygon into another, such that for every polygon the number of applicable transformations is bounded polynomially. Then the algorithm does a *random walk* on the *transformation graph*, which is the directed graph whose vertices are the polygons on S and where two vertices are connected by a directed edge whenever there is a transformation that turns one polygon into the other. If the transformation graph is connected, one can choose transition probabilites such that the distribution of the random walk converges to the uniform distribution.

The simplest transformation is the 2-flip. It removes two edges of the polygon and adds two other edges in order to get a new polygon. Unfortunately, there are point sets where the flip graph is not connected [4], which is shown by the existence of an isolated vertex.

For the k-flip, $k \geq 3$, it is unknown whether the corresponding flip graph is always connected. Hernando et al. [3] give a good overview of different types of flips and of subclasses of polygons whose flip graphs are connected.

2 Stretches and Twangs

Damian et al. [2] imagine the polygon as an elastic band that is attached to the points in S at its vertices. They introduce a new transformation, called *forward move*, based on the idea of deforming this elastic band. It consists of two different types of atomic operations, namely *stretches* and *twangs*.

Polygonal Wrap Neither stretches nor twangs are simplicity preserving. They produce an object called *polygonal wrap*, which does not have any proper crossings but can be self-touching.

Definition 1 ([2]) *A* polygonal wrap W *on* S *of length* $m \geq n$ *is a cyclic polygonal chain such that*

(W1) The wrap only bends at points of S*.*

(W2) Every point in S *is visited at least once.*

(W3) The wrap does not contain any proper crossings, i.e., there exists an arbitrarily small perturbation of the vertices of W *that makes the cyclic polygonal chain non-self-intersecting.*

*This is a short version of a master thesis with the same title submitted to the Institute of Computer Science at the University of Würzburg in 2016. The full version is available at http://www1.pub.informatik.uni-wuerzburg.de/pub/theses/2016-sering-master.pdf

If a point is visited more than once, we call it a point in multiple contact, *and a subsequence* (a, b, a) *is a* needle-pin *at* b. *Furthermore, we say that a line segment* ab does not properly cross W *if there is an arbitrarily small perturbation of* W *and* ab *such that* W *and* ab *do not intersect. For a subsequence* (a, b, c), *we call the cone in the minor arc the* convex side *and its complement the* reflex side.

Twang A twang is an operation that transforms one polygonal wrap into another. Informally, we choose a point in multiple contact that is not a needle-pin, detach one contact from the point and let the band snap back. The snapping band does not cross other vertices but attaches to them instead; see Figure 1a.

Definition 2 ([2]) *The operation* $\mathbf{Tw}(abc)$ *is defined for a subsequence* (a, b, c) *of a polygonal wrap* W *whenever the following three conditions hold:*

(T1) b *is in multiple contact.*

(T2) b *is not a hairpin.*

(T3) (a, b, c) *does not surround any other visits of* b.

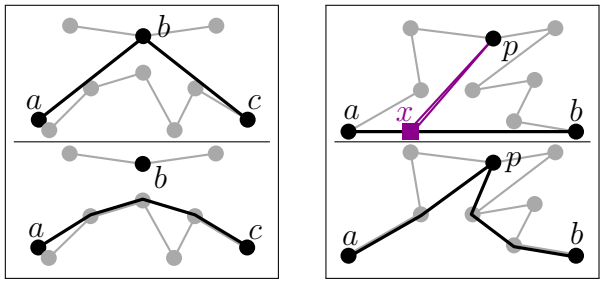
Then $\mathbf{Tw}(abc)$ *replaces the subsequence* (a, b, c) *in* W *by* $\mathbf{sp}(abc)$, *where* $\mathbf{sp}(abc)$ *is the shortest path from* a *to* c *inside of the triangle* $\triangle abc$ *that does not properly cross* W.

As long as there is at least one point in multiple contact, we can apply a twang to the wrap. Furthermore, a polygonal wrap without any points in multiple contact is a polygon. Although a twang might produce more multiple contacts in the process, we will show that repeated twanging will in fact terminate and restore a polygon.

Stretch Informally, a stretch is the operation of taking an edge e of the elastic band and attaching it to a point p. Similar to a twang, the band does not cross other points but instead wraps around them; see Figure 1b.

Definition 3 ([2]) *Given a polygonal wrap* W, *we say an edge* e *of* W *is* visible *to a point* p *if there is a point* x *in the interior of* e *such that the line segment* px *does not properly cross* W. *The point* x *is called the* spotted point. *The operation* $\mathbf{St}(e, p)$ *is defined for any edge* $e = (a, b)$ *of* W *and any vertex* $p \in S$ *if* e *is visible to* p. *To execute* $\mathbf{St}(e, p)$, *we replace* (a, b) *by* $(\mathbf{sp}(axp), \mathbf{sp}(pxb))$, *where* x *is the spotted point.*

In other words, we first add the spotted point x as a pseudo-vertex to the polygonal wrap and replace (a, b) by (a, x, p, x, b). Afterwards, we twang at x twice such that x can be removed again.



(a) Twanging at b: (a, b, c) is replaced by $sp(abc)$.

(b) Stretching (a, b) to p. Add (x, p, x) temporarily and twang at x twice.

Figure 1: Twang and Stretch.

A stretch is the main tool to modify a polygon and turn it into another polygon. But since it always creates at least one multiple contact, we need to apply a sequence of twangs in order to obtain a polygon.

Twang Cascade and Forward Move Next we define the transformation for the random walk.

Definition 4 ([2]) *Given a polygon* P *with an edge* e *and a point* p *such that* p *can see* e *and the spotted point* x *lies on the reflex side of the visit* (u, p, v), *a* forward move *consists of a stretch* $\mathbf{St}(e, p)$ *followed by a so-called* twang cascade, *which starts with the twang* $\mathbf{Tw}(upv)$ *and repeatedly twangs as long as there are vertices in multiple contact.*

In order to guarantee reversibility of the transformation, we also consider the time-reversal of a forward move and call it *reverse move*.

Pocket Reduction The main result of Damian et al. states the following:

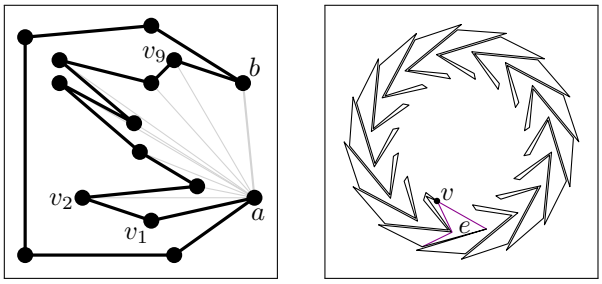
Theorem 1 *The transformation graph combining forward and reverse moves is connected and has a diameter of* $O(n^2)$. *Each node has degree* $\omega(n)$.

The key idea is to transform every polygon to a canonical polygon (see Figure 2a) by reducing one pocket after the other until there is only one non-reduced pocket left. This can be done by $O(n^2)$ forward moves.

3 Upper Bound on the Length of Twang Cascades

Damian et al. did not give a combinatorial upper bound on the length of a twang cascade and, therefore, no polynomial bound on the running time of a forward move is known.

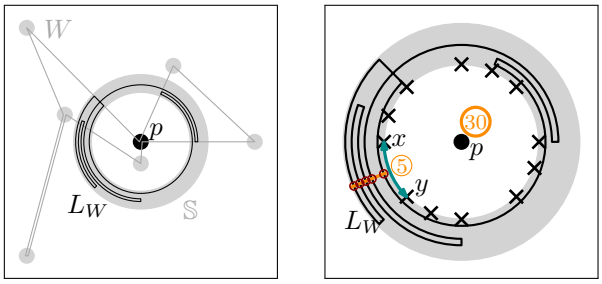
They showed, however, that there are forward moves that have twang cascades of length $\Theta(n^2)$ as shown in Figure 2b. In the following we will solve this



(a) Canonical polygon P_a with non-reduced pocket ab and vertices occurring clockwise around a.

(b) $\mathbf{St}(e, v)$ initiates a forward move twanging multiple times around the center.

Figure 2: Canonical polygon and quadratic twang cascade.



(a) The radial loop L_W is created by projecting the polygonal wrap W to the unit circle.

(b) $[xy]$ appears five times in L_W.
Hence $\mathbf{c}_W([xy]) = 5$.

Figure 3: Radial loop and coins.

open question and show that the length of a twang cascade is bounded by $O(n^3)$.

Given a set S of $n > 2$ points in the plane in general position and a polygonal wrap W that is created by a stretch on a polygon, we fix a point $p \in S$ and give an upper bound on the number of twangs that can occur at p in a twang cascade. We will show that this number is bounded by $O(n^2)$.

Markers, Elementary Arcs and Radial Loop For simplicity, consider p to be at the origin, that is, $p = (0, 0)$. First, we radially project every point in $S\backslash\{p\}$ to the unit circle.

Definition 5 *Consider the* radial projection*:*

$$\Pi\colon \mathbb{R}^2\backslash\{(0,0)\} \to \mathbb{S}, \quad q \mapsto \frac{q}{\|q\|}$$

Here $\mathbb{S}$ is the unit circle. We define

$$S^+ := \Pi(S\backslash\{p\}), \quad S^- := -S^+, \quad S^\pm := S^+ \cup S^-.$$

We call the elements of $S^\pm$ markers*. For each point $q \in S\backslash\{p\}$, we write q^+ for $\Pi(q)$ and q^- for $-\Pi(q)$. The bijective map $\iota : \mathbb{S} \to \mathbb{S}$ with $x \mapsto -x$ is called* central inversion*.*

The arcs between two neighbouring markers play an important role throughout this section.

Definition 6 *Let $x, y \in S^\pm$ be two markers. If the minor arc between x and y does not contain any other marker of $S^\pm$, we call it an* elementary arc *and write $[xy]$. Let Λ be the set of all elementary arcs.*

Since we assume that S is in general position, S^+ and S^- are disjoint, and therefore $|S^\pm| = 2n-2$. This is also the number of elementary arcs. The central inversion induces a bijection on the markers $q^+ \mapsto q^-$ and $q^- \mapsto q^+$ for all $q \in S\backslash\{p\}$ and, therefore, a bijection on the set of elementary arcs Λ as well:

$$[xy] \qquad \mapsto \qquad [\iota(x)\iota(y)]$$

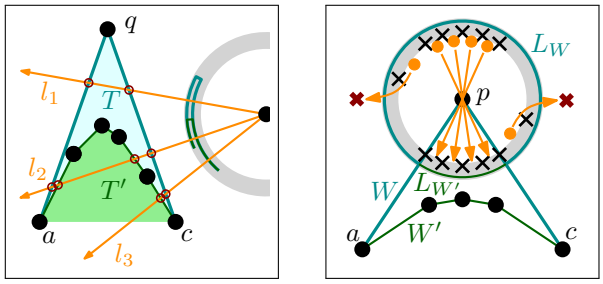
Next, we want to count how often the polygonal wrap goes around p in specific directions. This is well defined except for the spots where the wrap goes through p. For our purpose it is important to handle these cases as if the wrap went around the reflex side of p. We radially project the polygonal wrap to the unit circle in the following way.

Definition 7 *The* radial projection *of an edge (q, r) of W with $q \neq p$ and $r \neq p$ is the sequence of all elementary arcs between q^+ and r^+ on the convex side. For a subsequence (q, p, r) in W, the* radial projection *is the sequence of the elementary arcs between q^+ and r^+ on the major arc. If we incrementally replace every edge of W by its radial projection, we get the* radial loop *L_W. This is visualized in Figure 3a.*

Coin System Next, we introduce an integral potential, called coins. We assign as many coins to an elementary arc as often as the radial loop winds around p in the direction of this elementary arc; see Figure 3b. Whenever we twang at p, we remove at least two coins. Since the current number of coins is always non-negative, the number of twangs at p is bounded by the initial number of coins divided by two.

Definition 8 *For an elementary arc $[xy]$ let $\mathbf{c}_W([xy])$ be the number of times $[xy]$ appears in the radial loop L_W. We assign $\mathbf{c}_W([xy])$ many coins to $[xy]$. Furthermore, let $\mathbf{c}_W(p)$ be the total number of coins on the unit circle, that is,*

$$\mathbf{c}_W(p) := \sum_{[xy]\in\Lambda} \mathbf{c}_W([xy])$$



(a) l_1 crosses T but not T'. l_2 crosses both twice. l_3 crosses both once.

(b) The coin movement if we twang at p.

Figure 4: A twang only decreases the number of coins.

The Combinatorial Upper Bound In this section we want to prove the combinatorial upper bound on the length of twang cascades by bounding the number of twangs at each point p.

The following theorem states that the number of coins does not increase when we execute a twang.

Theorem 2 *A twang $W \to W'$ at $q \in S \backslash \{p\}$ never increases the number of coins at p. In other words, $\mathbf{c}_{W'}(p) \leq \mathbf{c}_W(p)$.*

The proof uses the fact that the twanged subsequence T' is in convex position and lies inside the original subsequence T; see Figure 4a. If T' goes around p in the direction of an elementary arc then so did T. Therefore, the number of coins on each elementary arc can only decrease. Note that a special case occurs when p is a neighbour of q. We omit the details.

The next theorem considers the case of twanging at p. In order to bound the number of twangs by the number of coins, we need to make sure that the number of coins decreases each time we twang.

Theorem 3 *Every twang $W \to W'$ at p decreases the number of coins at p by at least two. In other words, $\mathbf{c}_{W'}(p) \leq \mathbf{c}_W(p) - 2$.*

The radial projection of the original subsequence lies on the major arc, whereas the radial projection of the twanged subsequence is on the minor arc. One can imagine that every elementary arcs $[xy]$ on the major arc gives one of their coins to $\iota([xy])$ on the opposite side, but only if $\iota([xy])$ is on the minor arc. At least two elementary arcs cannot give one of their coins to their opposite elementary arc and therefore these coins are removed from the system. How the coins move is illustrated in Figure 4b.

In the next step we give an upper bound on the number of coins on each point p at the beginning of a twang cascade.

Lemma 4 *Let P be a polygon on S, and $p \in S$. Then $\mathbf{c}_P(p) \leq n^2$.*

This is true due to the fact that each of the n edges can at most contribute one coin to at most n elementary arcs.

Lemma 5 *A stretch does not add more than $O(n)$ coins to p.*

This holds because a stretch means basically to add two edges followed by some twangs.

In summary, there are no more than $O(n^2)$ coins on p at the beginning of a twang cascade. Therefore, only $O(n^2)$ twangs can occur at p. Summing over all n points we receive our final result:

Theorem 6 *In every forward move, the length of a twang cascade is bounded by $O(n^3)$.*

If we count more carefully and consider a lower bound of coins for polygons, we can show that the length of a twang cascade is in fact bounded by $n^3/2$.

4 Open Problems

We still need to solve several problems to obtain an efficient random generator for polygons.

1. Can reverse moves be computed efficiently?
2. Is the transformation graph still connected if we allow only forward moves?
3. Is the random walk rapidly mixing?

We assume the first question to be answered with no and the last with yes.

References

[1] T. Auer and M. Held. RPG – heuristics for the generation of random polygons. *Proc. 8th Canad. Conf. Comput. Geom (CCCG'98)*, pages 38–44, 1998.

[2] M. Damian, R. Flatland, J. O'Rourke, and S. Ramaswami. Connecting polygonizations via stretches and twangs. *Theory of Computing Systems*, 47(3):674–695, 2010.

[3] C. Hernando, M. E. Houle, and F. Hurtado. On local transformation of polygons with visibility properties. *Theoretical Computer Science*, 289(2):919–937, 2002.

[4] M. E. Houle. On local transformation of polygons. In *Proc. Computing: 2nd Australasian Theory Symp. (CATS'96)*, volume 18 of *Australian Comput. Sci. Commun.*, pages 64–71, 1996.

[5] C. Zhu, G. Sundaram, J. Snoeyink, and J. S. Mitchell. Generating random polygons with given vertices. *Computational Geometry*, 6(5):277–290, 1996.

Is Area Universality $\forall\exists\mathbb{R}$-complete?

Linda Kleist * Tillmann Miltzow † Paweł Rzążewski †‡

Abstract

We say a plane graph G' is a *redrawing* of another plane graph G if they have the same face structure. Given a *face area assignment* of all inner faces of G, we are looking for a straight-line redrawing of G realizing the prescribed face areas. If it holds that for every face area assignment there exists an appropriate redrawing of G, then G is *area-universal*. By Area Universality we denote the algorithmic problem of deciding if G is area-universal. We introduce and motivate the natural complexity class $\forall\exists\mathbb{R}$ and conjecture that Area Universality is $\forall\exists\mathbb{R}$-complete. We also show that some variants of Area Universality are $\exists\mathbb{R}$-complete.

1 Introduction

Certain geometric problems are closely linked to real algebra. One simple example is Pappus's Hexagon Theorem (see Figure 1).

Theorem 1 (Pappus's Hexagon Theorem) *Let A, B, C be three points on a straight line and let X, Y, Z be three points on another line. If the lines $\overline{AY}, \overline{BZ}, \overline{CX}$ intersect the lines $\overline{BX}, \overline{CY}, \overline{AZ}$, respectively, then the three points of intersection are collinear.*

Although the statement is intrinsically geometric, it is non-trivial to prove and most known proofs have some algebraic flavor, see [8, Chapter 1].

The complexity class $\exists\mathbb{R}$ consists of all problems that reduce in polynomial time to the problem denoted as Existential Theory of the Reals (ETR), that has as input a first order formula over the reals in prenex form that contains only existential quantifiers,

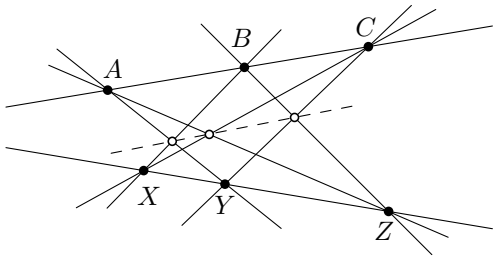


Figure 1: Illustration of Pappus's Hexagon Theorem.

and asks whether the formula is true or not. An example of such a formula is:

$$\exists x\, \exists y : x^2 + y^2 > 1 \wedge 3x + 2y = 10.$$

The importance of $\exists\mathbb{R}$ stems from the fact that many natural geometric problems are $\exists\mathbb{R}$-complete.

A graph is called a *segment graph* if it can be represented as the intersection graph of segments in the plane. The recognition problem of segment graphs is a prominent geometric problem that is $\exists\mathbb{R}$-complete, see [7]. $\exists\mathbb{R}$-completeness reflects the deep algebraic connections between segment graphs and real algebra. This connection has two important consequences that we must be aware of. The first consequence is that there is little hope to find a simple combinatorial algorithm recognizing segment graphs, as simple combinatorial algorithms to decide ETR are not known. In fact, without certain algebraic tools, which bound the length of integer coordinate representations, we would not even know if the recognition problem is *decidable*. As a second consequence, this graph class is also difficult to study from a combinatorial point of view. This last statement should be understood more as an opinion than a provable fact. However, it is not just difficult to decide whether a given graph is a segment graph. It requires algebraic arguments eventually.

Let us define Area Universality formally, before we proceed with the discussion.

Area Universality

Input: Plane graph $G = (V, E)$.
Question: Does there exist an area-realizing straight-line redrawing of G, for every possible face area assignment?

*Institut für Mathematik, Technische Universität Berlin (TU), Berlin, Germany kleist@math.tu-berlin.de

†Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI), Budapest, Hungary t.miltzow@gmail.com p.rzazewski@mini.pw.edu.pl
Supported by the ERC grant PARAMTIGHT: "Parameterized complexity and the search for tight complexity results", no. 280152.
‡Faculty of Mathematics and Information Sciences, Warsaw University of Technology, Warsaw, Poland

Geometric problems that are $\exists\mathbb{R}$-complete usually ask for existence of certain objects satisfying some given properties. However, the nature of Area Universality seems to be different. We therefore define the new complexity class $\forall\exists\mathbb{R}$ as the set of all problems that reduce in polynomial time to Universal Existential Theory of the Reals (UETR). The input of UETR is a first order formula over the reals in prenex form, which starts with a block of universal quantifiers followed by a block of existential quantifiers and is otherwise quantifier-free. We ask if the formula is true. An example of such a formula is:

$$\forall x\, \forall y\, \exists z : x^2 + z^2 \geq 1 \wedge 3x + 2y = 10z.$$

It is easy to observe and well-known that NP is contained in $\exists\mathbb{R}$ and it is even simpler to see that $\exists\mathbb{R}$ is contained in $\forall\exists\mathbb{R}$. In the same way that NP $\subseteq \exists\mathbb{R}$, we can observe that Π_2^p, the complexity class on the second level of the polynomial time hierarchy (see [2, Chapter 5]), is contained in $\forall\exists\mathbb{R}$. Highly non-trivial is the containment of $\forall\exists\mathbb{R}$ in PSPACE, see [3]. For all we know, all these complexity classes could collapse, as we do not know whether NP and PSPACE constitute two different or the same complexity class, see Figure 2. Analogously to $\exists\mathbb{R}$ being sometimes referred to as the real counterpart to NP, $\forall\exists\mathbb{R}$ can be regarded as the real counterpart to Π_2^p. Thus $\exists\mathbb{R} \neq \forall\exists\mathbb{R}$ can be believed with similar confidence as NP $\neq \Pi_2^p$. In addition, the expressibility of $\forall\exists\mathbb{R}$ is larger than $\exists\mathbb{R}$ in an algebraic sense, see [5].

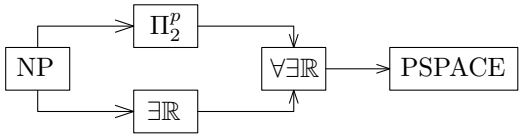


Figure 2: Containment relations of complexity classes.

The membership of Area Universality in $\forall\exists\mathbb{R}$ can be seen by reducing the problem to UETR. We use the block of universal quantifiers to describe the face area assignment and the block of existential quantifiers to describe the placement of the vertices of the redrawing of G; the quantifier-free formula checks if the placement indeed realizes a straight-line redrawing of G with the specified areas.

Conjecture Area Universality *is* $\forall\exists\mathbb{R}$*-complete.*

While this conjecture, if true, would show that Area Universality is a really difficult problem in an algebraic and combinatorial sense, it would also give the first known natural geometric problem that is complete for $\forall\exists\mathbb{R}$.

To stimulate the research on our conjecture, we want to point out that we are not capable to determine if

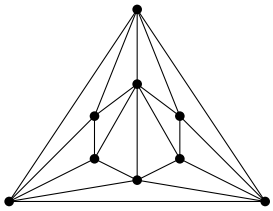


Figure 3: Is this graph area-universal?

the small graph G in Figure 3 is area-universal. We offer a personalized super cool mug to the first person who proves or disproves area-universality of G.

Little is known about area-universal graphs. It is very straightforward to observe that stacked triangulations are area-universal. Clearly, if a graph is area-universal, every subgraph of it is also area-universal. This implies for instance that outerplane graphs are area-universal. The only non-trivial class of graphs that are known to be area-universal are cubic plane graphs [10] and all subdivisions of plane graphs [6].

On the negative side, it was shown by Ringel [9] that the octahedron graph is not area-universal. Kleist [6] showed recently that all Eulerian triangulations and the icosahedron graph are not area-universal.

Our Results. As a first step towards proving our conjecture, we consider two variants of Area Universality.

Planar Prescribed Area*

Input: Plane graph $G = (V, E)$, a face area assignment on the set F of inner faces $\mathcal{A}\colon F \to \mathbb{R}^+$, fixed positions for a subset of vertices $V' \subset V$.
Question: Does there exist an area-realizing straight-line redrawing of G respecting the given positions for all $v \in V'$?

The first result concerns the hardness of Planar Prescribed Area*.

Theorem 2 Planar Prescribed Area* *is* $\exists\mathbb{R}$*-complete.*

Theorem 2 was foreshadowed by Biedl et al. [4], who showed examples of plane graphs along with integer face area assignments, such that every area-realizing redrawing requires irrational coordinates of vertices.

The second result concerns a related problem, in which we drop the planarity restriction.

Prescribed Area in Hypergraphs

Input: A 3-uniform hypergraph $G = (V, E)$ and an area assignment $\mathcal{A}$ for each edge $e \in E$.

Question: Does there exist a placement of the vertices V such that the area of each triangle $e \in E$ is exactly $\mathcal{A}(e)$?

Note that we have no non-crossing constraint in Prescribed Area in Hypergraphs.

Theorem 3 Prescribed Area in Hypergraphs *is* $\exists\mathbb{R}$*-complete.*

Due to space constraints, we skip the proof of Theorem 3. The main idea is to use gadgets simulating von Staudt constructions [7]. It is easy to build these gadgets, because we can force points to lie on parallel lines.

2 Planar Prescribed Area*

Recently Abrahamsen et al. showed that the following variant of ETR is $\exists\mathbb{R}$-complete [1].

ETR-inv

Input: $\Phi = \exists x_1 \dots x_n : \varphi(x_1, \dots, x_n)$, where φ is a conjunction of constraints of the following form: $x = 1$, $x + y = z$, $x \cdot y = 1$.

Question: Is Φ true?

They showed that it is sufficient to consider instances, which are either negative, or have a solution within range of $(0, 10^5)$.

Proof of Theorem 2 (sketch). We reduce from ETR-inv. Given $\Phi = \exists x_1 \dots x_n : \varphi(x_1, \dots, x_n)$, we construct a plane graph G_Φ, give a face area assignment $\mathcal{A}$ and a subset of vertex positions, such that G_Φ has a realizing drawing if and only if φ is satisfiable by real values from the interval $(0, 10^5)$.

To do so, we consider the bipartite incidence graph of variables and constraints and fix an orthogonal drawing on an integer grid. For an example see Figure 4. We need six types of gadgets: vertices for variables, splitter, inversion and addition constraints, as well as wires and crossings of wires. However, a crossing gadget can be simulated by linking three addition gadgets appropriately. Some of the vertices used in our gadgets will have prescribed positions. We call such vertices *fixed* and all other vertices *flexible*.

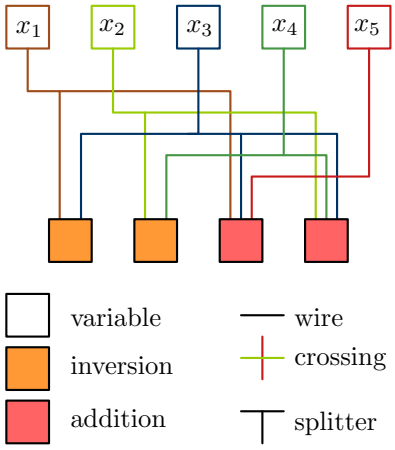


Figure 4: The incidence graph of $\varphi = (x_1 \cdot x_3 = 1) \wedge (x_2 \cdot x_3 = 1) \wedge (x_1 + x_3 = x_5) \wedge (x_2 + x_3 = x_4)$.

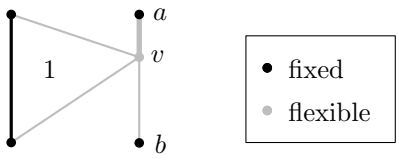


Figure 5: The variable gadget.

The *variable gadget* consists of a triangle with prescribed area 1, where two vertices are already fixed (see Fig. 5). In any realizing drawing the third vertex v of the triangle, which is flexible, lies on a line. Moreover, we can force it to lie on a specific segment, bounded by two fixed vertices a, b. The distance of v to a represents the *scaled* value of the variable x. The true value is obtained by multiplying with 10^5. For this reduction to work, it is important that we can bound the range of each variable. In the following we sometimes assume that a flexible vertex is forced to lie on a specified line. This is realized by a proper use of a variable gadget.

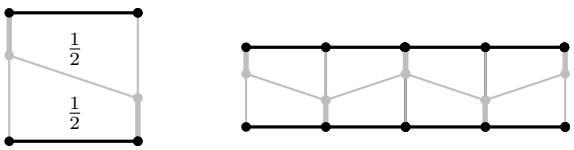


Figure 6: The wire gadget.

The *wire gadget* consists of several box-like fragments: Four fixed vertices positioned at the corners of a square of area 1 and two opposite fixed edges (see Fig. 6). Each of the other two sides of the square is subdived by a flexible vertex which are joined by an edge. Each of the two faces has a prescribed area of $\frac{1}{2}$. Note that if one of the flexible vertices is collinear with its fixed neighbors, so is the other one. Moreover, the segment representing the value of the variable changes

from top to bottom or vice versa. Hence, if necessary, we may use an odd number of fragments in order to invert the side where the value is represented.

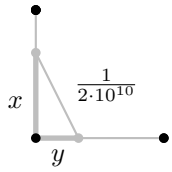


Figure 7: The inversion gadget.

The *inversion gadget* consists of three fixed vertices, two flexible vertices and a triangular face (see Fig. 7). Given input x, y, it enforces $x \cdot y = 1$. We can assume that both flexible vertices are enforced to lie on the lines spanned by its fixed neighbors. Hence, one variable represents the length of the base and the other the length of the height of the triangular face. If the prescribed area of $\frac{1}{2\cdot 10^{10}}$ is realized, the two values are inverses of each other.

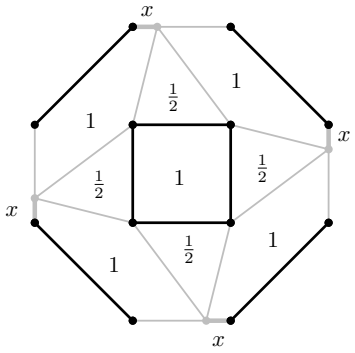


Figure 8: The splitter gadget.

The *splitter gadget* contains a central fixed square of area 1, whose each side is adjacent to a triangle of area $\frac{1}{2}$ (see Fig. 8). Each triangle fixes a flexible vertex on a line. The idea is that the value of one variable fixes the value of all variables by pushing the area circularly. Note that each face of area 1 has exactly two flexible vertices. If one is fixed, clearly the other must achieve the correct area. Since it must lie on a line, its position is uniquely determined. We also use splitter gadgets to realize turns.

The *addition gadget* is the most complex (see Fig. 9). It consists of many already fixed vertices and six flexible ones, one for each of the values x, y, and $x+y$, and three in the interior. One of the inner flexible vertices is adjacent to 4 regions, and its position is determined by the 2 values $-(x+y)$ and y. The idea is that it can transport y from right to left and $-(x+y)$ from top to bottom. Thanks to this, we can use the fact that $x+y$ and $-(x+y)$ can cancel out within the gadget. ■

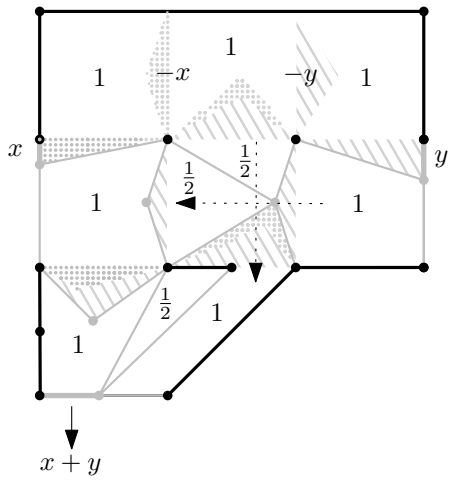


Figure 9: The addition gadget.

References

[1] Mikkel Abrahamsen, Anna Adamaszek, and Tillmann Miltzow. The art gallery problem is $\exists\mathbb{R}$-complete. *in preparation*, 2017.

[2] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[3] Saugata Basu, Richard Pollack, and Marie-Françoise Roy. *Algorithms in real algebraic geometry*. Springer-Verlag, 2006.

[4] Therese C. Biedl and Lesvia Elena Ruiz Velázquez. Drawing planar 3-trees with given face areas. *Comput. Geom.*, 46(3):276 285, 2013.

[5] James H. Davenport and Joos Heintz. Real quantifier elimination is doubly exponential. *Journal of Symbolic Computation*, 5(1):29 35, 1988.

[6] Linda Kleist. Drawing planar graphs with prescribed face areas. In *WG 2016 Proc., LNCS 9941*, pages 158 170. Springer, 2016.

[7] Jiří Matoušek. Intersection graphs of segments and $\exists\mathbb{R}$. *CoRR, abs/1406.2636*, 2014.

[8] Jürgen Richter-Gebert. *Perspectives on projective geometry: a guided tour through real and complex geometry*. Springer Science & Business Media, 2011.

[9] Gerhard Ringel. Equiareal graphs. *Contemporary Methods in Graph Theory*, pages 503 505, 1990.

[10] Carsten Thomassen. Plane cubic graphs with prescribed face areas. *Combinatorics, Probability & Computing*, 1(371-381):2 10, 1992.

A Lower Bound for the Dynamic Conflict-Free Coloring of Intervals with Respect to Points

Mark de Berg* Tim Leijsen* Aleksandar Markovic* André van Renssen†‡
Marcel Roeloffzen† ‡ Gerhard Woeginger*

Abstract

We introduce the dynamic conflict-free coloring problem for a set S of intervals in $\mathbb{R}^1$ with respect to points, where the goal is to maintain a conflict-free coloring for S under insertions and deletions. We investigate trade-offs between the number of colors used and the number of intervals that are recolored upon insertion or deletion of an interval. We provide a lower bound on the number of recolorings as a function of the number of colors, which implies that with $O(1)$ recolorings per update the worst-case number of colors is $\Omega(\log n/\log\log n)$, and that any strategy using $O(1/\varepsilon)$ colors needs $\Omega(\varepsilon n^{\varepsilon})$ recolorings. We also provide a stronger lower bound against so-called local algorithms.

1 Introduction

Consider a set S of fixed base stations that can be used for communication by mobile clients. Each base station has a transmission range, and a client can potentially communicate via that base station when it lies within the transmission range. However, when a client is within reach of several base stations that use the same frequency, the signals will interfere. Hence, the frequencies of the base stations should be assigned in such a way that this problem does not arise. Moreover, the number of used frequencies should not be too large. Even *et al.* [6] and Smorodinsky [10] introduced conflict-free colorings to model this problem, as follows. Let S be a set of disks in the plane, and for a point $q \in \mathbb{R}^2$ let $S(q) \subseteq S$ denote the set of disks containing the point q. A coloring of the disks in S is *conflict-free* if, for any point $q \in \mathbb{R}^2$, the set $S(q)$ has at least one disk with a color that is unique among the disks in $S(q)$. Even *et al.* [6] proved that any set of n disks in the plane admits a conflict-free coloring with $O(\log n)$ colors, and this bound is tight in the worst case.

The concept of conflict-free colorings can be generalized and extended in several ways, giving rise to a host of challenging problems. Below we mention some of them; a more extensive overview is given by Smorodinsky [11]. One obvious generalization is to work with types of regions other than disks [6, 8]. One can also consider the inverse setting, where one wants to color a given set P of n points in the plane, such that any disk—or rectangle, or other range from a given family—contains at least one point with a unique color (if it contains any point at all). This too was studied by Even *et al.* [6] and they show that this can be done with $O(\log n)$ colors when the ranges are disks or scaled translations of a single centrally symmetric convex polygon.

These results deal with the static setting, in which the set of objects to be colored is known in advance. This may not always be the case, leading Fiat *et al.* [7] to introduce the *online* version of the conflict-free coloring problem. Here the objects to be colored arrive one at a time, and each object must be colored upon arrival. Fiat *et al.* show that when coloring points in the plane with respect to disks, n colors may be needed in the online version. Hence, they turn their attention to the 1-dimensional problem of online coloring points with respect to intervals. They prove that this can be done deterministically with $O(\log^2 n)$ colors and randomized with $O(\log n \log\log n)$ colors with high probability. Later Chen [4] gave a randomized algorithm that uses $O(\log n)$ colors with high probability. Similar results were also obtained for conflict-free colorings of points with respect to halfplanes, unit disks and axis-aligned rectangles of almost the same size, using $O(\text{polylog } n)$ colors with high probability. Bar-Noy *et al.* [1] considered the case where recolorings are allowed for each insertion. They prove that for coloring points in the plane with respect to halfplanes, one can obtain a coloring with $O(\log n)$ colors in an online setting at the cost of $O(n)$ recolorings in total. More recent variants include strong conflict-free colorings [3, 9], where we require several unique colors, and conflict-free multicolorings [2], which allow assigning multiple colors to a point.

Our contributions. We introduce a variant of the conflict-free coloring problem where the objects to be colored arrive and disappear over time. This *dynamic conflict-free coloring problem* models a scenario

*TU Eindhoven. MdB, AM and GW are supported by the Netherlands' Organisation for Scientific Research (NWO) under project no. 024.002.003.
†National Institute of Informatics, Tokyo, Japan
‡JST, ERATO, Kawarabayashi Large Graph Project

where new base stations may be deployed (to deal with increased capacity demands, for example) and existing base stations may break down or be taken out of service (either permanently or temporarily). This natural variant has, to the best of our knowledge, not been considered so far. It is easy to see that, unless one maintains a regular coloring in which any two intersecting objects have distinct colors, there is always a sequence of deletions that invalidates a given conflict-free coloring. Hence, recolorings are needed to ensure that the new coloring is conflict-free. This leads to the question: how many recolorings are needed to maintain a coloring with a certain number of colors? We initiate the study of dynamic conflict-free colorings by considering the problem of coloring intervals with respect to points. In this variant, we are given a (dynamic) set S of intervals in $\mathbb{R}^1$, which we want to color such that for any point $q \in \mathbb{R}^1$ the set $S(q)$ of intervals containing q contains an interval with a unique color. In the static setting, coloring intervals is rather easy: a simple procedure yields a conflict-free coloring with three colors. The dynamic version turns out to be much more challenging.

We prove lower bounds on the possible tradeoffs between the number of colors used and the worst-case number of recolorings per update: for any algorithm that maintains a conflict-free coloring on a sequence of n insertions of intervals with at most $c(n)$ colors and at most $r(n)$ recolorings per insertion, we must have $r(n) > n^{1/(c(n)+1)}/(8c(n))$. This implies that for $O(1/\varepsilon)$ colors we need $\Omega(\varepsilon n^{\varepsilon})$ recolorings, and with only $O(1)$ recolorings we must use $\Omega(\log n/\log\log n)$ colors. In the extended version of this paper [5] we present several algorithms that achieve bounds close to our lower bound.

2 Lower bounds for semi-dynamic conflict-free colorings

In this section we present lower bounds on the semi-dynamic (insertion only) conflict-free coloring problem for intervals. More precisely, we present lower bounds on the number of recolorings necessary to guarantee a given upper bound on the number of colors. We prove a general lower bound and a stronger bound for so-called local algorithms. The general lower bound uses a construction where the choice of segments to be added depends on the colors of the segments already inserted. This adaptive construction is also valid for randomized algorithms, but it does not give a lower bound on the expected behavior.

Theorem 1 *Let* ALG *be an insertion-only deterministic algorithm for the dynamic conflict-free coloring of intervals. Suppose that on any sequence of $n > 0$ insertions,* ALG *uses at most $c(n)$ colors and $r(n)$ recolorings per insertion, where $r(n) > 0$. Then $r(n) > n^{1/(c(n)+1)}/(8c(n))$.*

Proof. We first fix a value for n and define $c := c(n)$ and $r := r(n)$. Our construction will proceed in rounds. In the i-th round we insert a set R_i of n_i disjoint intervals—which intervals we insert depends on the current coloring provided by ALG. After R_i has been inserted (and colored by ALG), we choose one of the colors used by ALG for R_i to be the *designated color* for the i-th round. We denote this designated color by c_i. We will argue that in each round we can pick a different designated color, so that the number of rounds, ρ, is a lower bound on the number of colors used by ALG. We then prove a lower bound on ρ in terms of n, c, and r, and derive the theorem from the inequality $\rho \leqslant c$.

To describe our construction more precisely, we need to introduce some notation and terminology. Let $R_i := \{I_1, \ldots, I_{n_i}\}$, where the intervals are numbered from left to right. (Recall that the intervals in R_i are disjoint.) To each interval $I = I_j$ we associate the set $I^e := (a, b)$, where a is the right endpoint of I, and b is the left endpoint of I_{j+1} if $j < n_i$ and $+\infty$ if $j = n_i$, that is, I^e represents the empty space to the right of I. We call (I, I^e) an *i-brick*. We define the color of a brick (I, I^e) to be the color of I, and we say a point or an interval is contained in this brick if it is contained in $I \cup I^e$. Recall that each round R_i has a designated color c_i. We say that an i-brick $B := (I, I^e)$ is *living* if the following two conditions are met:

- I has the designated color c_i;
- if $i > 1$ then both I and I^e contain living $(i-1)$-bricks.

A brick that is not alive is called *dead* and an event such as a recoloring that causes a brick to become dead is said to *kill* the brick. By recoloring an interval I, ALG can kill the brick $B = (I, I^e)$ and the death of B may cause some bricks containing B to be killed as well.

We can now describe how we generate the set R_i of intervals we insert in the i-th round and how we pick the designated colors. (Note that the designated color of a round is fixed once it is picked; it is not updated when recolorings occur.) We denote by R_i^* the subset of intervals $I \in R_i$ such that (I, I^e) is a living i-brick. Note that R_i^* can be defined only after the i-th round, when we have picked the designated color c_i.

1. The set R_1 contains the $\frac{n}{2}$ intervals $[0,1], [2,3], \ldots, [n-2, n-1]$, and the designated color c_1 of the first round is the color used most often in the coloring produced by ALG after insertion of the last interval in R_1.
2. To generate R_i for $i > 1$, we proceed as follows. Partition R_{i-1}^* into groups of $4r$ consecutive intervals. (If $|R_{i-1}^*|$ is not a multiple of $4r$, the final

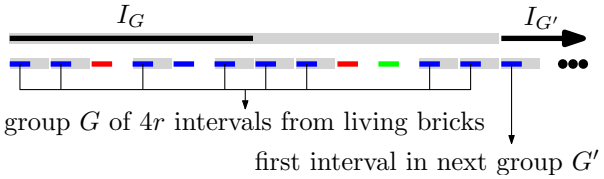


Figure 1: Example of how the intervals are created when $r = 2$. The designated color c_{i-1} is blue, and the grey rectangles indicate living $(i-1)$-bricks. The grey rectangle around I_G indicates the brick (I_G, I_G^e). Note that $I_{G'}$ extends further to the right.

group will be smaller than $4r$. This group will be ignored.) For each group $G := I_1, \ldots, I_{4r}$ we put an interval I_G into R_i, which starts at the left endpoint of I_1 and ends slightly before the left endpoint of I_{2r+1}; see Fig. 1 for an illustration.

The designated color c_i is picked as follows. Consider the coloring after the last interval of R_i has been inserted, and let $C(i)$ be the set of colors assigned by ALG to intervals in R_i and that are not a designated color from a previous round—we argue below that $C(i) \neq \emptyset$. Then we pick c_i as the color from $C(i)$ that maximizes the number of living i-bricks.

We continue generating sets R_i in this manner until $|R_i^*| < 4r$, at which point the construction finishes. Below we prove that in each round ALG must introduce a new designated color, and we prove a lower bound on the number of rounds in the construction.

Claim. *Let $B = (I, I^e)$ be a living i-brick. Then for any $j \in \{1, \ldots, i\}$ there is a point $q \in I \cup I^e$ that is contained in a single interval of color c_j and in no other interval from $\bigcup_{\ell=1}^{i-1} R_\ell$. Moreover, there is a point $q \in I \cup I^e$ not contained in any interval from $\bigcup_{\ell=1}^{i-1} R_\ell$.*
Proof. We prove this by induction on i. For $i = 1$ the statement is trivially true, so suppose $i > 1$. By definition, both I and I^e contain living $(i-1)$-bricks, $\overline{B}$ and $\overline{B}^e$. Using the induction hypothesis we can now select a point q with the desired properties: for $j = i$ we use that $\overline{B}$ contains a point that is not contained in any interval from $\bigcup_{\ell=1}^{i-1} R_\ell$, for $j < i$ we use that $\overline{B}^e$ contains a point in an interval of color c_j and in no other interval from $\bigcup_{\ell=1}^{i-1} R_\ell$, and to find a point not contained in any interval from $\bigcup_{\ell=1}^{i-1} R_\ell$ we can also use $\overline{B}^e$. □

Now consider the situation after the i-th round, but before we have chosen the designated color c_i. We say that a color c is *eligible* (to become c_i) if $c \neq c_1, \ldots, c_{i-1}$, and we say that an i-brick (I, I^e) is eligible if its color is eligible and (I, I^e) would be living if we were to choose its color as the designated color c_i. Note that due to some recolorings, some of the newly inserted intervals might not contain any living brick and hence can never be living no matter the designated color; the next claim shows that at most half intervals inserted this round share this property.

Claim. *Immediately after the i-th round, at least half of the i-bricks are eligible.*
Proof. Consider an i-brick (I, I^e). At the beginning of the i-th round, before we have actually inserted the intervals from R_i, both the interval I and its empty space I^e contain $2r$ living $(i-1)$-bricks. As the intervals from R_i are inserted, ALG may recolor certain intervals from $R_1 \cup \ldots \cup R_{i-1}$, thereby killing some of these $(i-1)$-bricks. Now suppose that ALG recolored at most $2r-1$ of the intervals from $R_1 \cup \ldots \cap R_{i-1}$ that are contained in $I \cup I^e$. Then both I and I^e still contain a living $(i-1)$-brick. By the previous claim, this implies ALG cannot use any of the colors c_j with $j < i$ for I. Hence, the color of I is eligible and the i-brick (I, I^e) is eligible as well.

It remains to observe that ALG can do at most rn_i recolorings during the i-th round. We just argued that to prevent an i-brick from becoming eligible, ALG must do at least $2r$ recolorings inside that brick. Hence, ALG can prevent at most half of the i-bricks from becoming eligible. □

Recall that after the i-th round we pick the designated color c_i that maximizes the number of living i-bricks. In other words, c_i is chosen to maximize $|R_i^*|$. Next we prove a lower bound on this number. Recall that ρ denotes the number of rounds.

Claim. *For $1 \leqslant i \leqslant \rho$ we have $|R_i^*| \geqslant n_1/(8rc)^i - 1$.*
Proof. Since ALG can use at most c colors, we have $|R_1^*| \geqslant n_1/c$. Moreover, for $i > 1$ the number of intervals we insert is $\lfloor |R_{i-1}^*|/4r \rfloor$. By the previous claim at least half of these are eligible. The eligible intervals have at most c different colors, so if we choose c_i to be the most common color among them we see that $|R_i^*| \geqslant \lfloor |R_{i-1}^*|/4r \rfloor /2c$. We thus obtain the following recurrence:

$$|R_i^*| \geqslant \begin{cases} \dfrac{\lfloor |R_{i-1}^*|/4r \rfloor}{2c} & \text{if } i > 1, \\ \dfrac{n_1}{c} & \text{if } i = 1. \end{cases} \tag{1}$$

We can now prove $|R_i^*| > \frac{n_1}{(8rc)^i} - 1$ using induction. □

Finally we can derive the desired relation between n, c, and r. Since $n_1 = n/2$ and $n_{i+1} < n_i/2$ for all $i = 1, \ldots, \rho - 1$, the total number of insertions is less than n. The construction finishes when $|R_i^*| < 4r$. Hence, ρ, the total number of rounds, must be such that $|R_\rho^*| = n/(2(8rc)^\rho) - 1 < 4r$, which implies $\rho > \log_{8rc}(n/(8r+2)) > \log_{8rc} n - 1$. The number

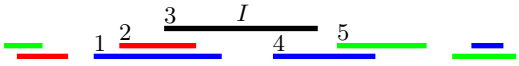


Figure 2: Example of a signature. The set $S(I)$ contains the segments labeled 1,2,4,5. The signature of I is $\langle 2, 1, 3, 4, 5, \text{red}, \text{blue}, \text{NIL}, \text{blue}, \text{green}\rangle$.

of colors used by ALG is at least ρ, since every round has a different designated color. Thus $c > \log_{8rc} n - 1$ and so $n \leqslant (8rc)^{c+1}$. □

Two interesting special cases of the theorem are the following: with $r = O(1)$ we will have $c = \Omega(\log n / \log\log n)$, and for $c = O(1/\varepsilon)$ (for some small fixed $\varepsilon > 0$) we need $r = \Omega(\varepsilon n^{\varepsilon})$. Note that the theorem requires $r > 0$. Obviously the $\Omega(\log n / \log\log n)$ lower bound on c that we get for $r = 1$ also holds for $r = 0$. For the special case of $r = 0$—this is the standard online version of the problem—we can prove a stronger result, however: here we need at least $\lfloor \log n \rfloor + 1$ colors. This bound even holds for a nested set of intervals, that is, a set S such that $I \subset I'$ or $I' \subset I$ for any two intervals $I, I' \in S$. We also show in the extended paper [5] that a greedy algorithm achieves this bound for nested intervals.

Local algorithms. We now prove a stronger lower bound for so-called local algorithms. Intuitively, these are deterministic algorithms where the color assigned to a newly inserted interval I only depends on the structure and the coloring of the connected component where I is inserted—hence the name *local*. More precisely, local algorithms are defined as follows.

Suppose we insert an interval I into a set S of intervals that have already been colored. The union of the set $S \cup \{I\}$ consists of one or more connected components. We define $S(I) \subseteq S$ to be the set of intervals from S that are in the same connected component as I. (In other words, if we consider the interval graph induced by $S \cup \{I\}$ then the intervals in $S(I)$ form a connected component with I.) Order the intervals in $S(I) \cup \{I\}$ from left to right according to their left endpoint, and then assign to every interval its rank in this ordering as its label. (Here we assume that all endpoints of the intervals in $S(I) \subseteq S$ are distinct. It suffices to prove our lower bound for this restricted case.) Based on this labeling we define a signature for $S(I) \cup \{I\}$ as follows. Let $\lambda_1, \ldots, \lambda_k$, where $k := |S(I)| + 1$, be the sequence of labels obtained by ordering the intervals from left to right according to their right endpoint. Furthermore, let c_i be the color of the interval labeled i, where $c_i = \text{NIL}$ if the interval labeled i has not yet been colored. Then we define the *signature* of $S(I) \cup I$ to be the sequence $\text{sig}(I) := \langle \lambda_1, \ldots, \lambda_k, c_1, \ldots, c_k \rangle$; see Fig. 2.

We now define a dynamic algorithm ALG to be *local* if upon insertion of an interval I the following holds: (i) ALG only performs recoloring in $S(I)$, and (ii) the color assigned to I and the recolorings in $S(I)$ are uniquely determined by $\text{sig}(I)$, that is, the algorithm is deterministic with respect to $\text{sig}(I)$. Note that randomized algorithms are not local.

To strengthen Theorem 1 for the case of local algorithms, it suffices to observe that the intervals inserted in the same round must all receive the same color. Hence, the factors c in the denominators of Inequality (1) disappear, leading to the theorem below. Note that for $r(n) = O(1)$, we now get the lower bound $c(n) = \Omega(\log n)$.

Theorem 2 *Let* ALG *be a local insertion-only algorithm for the dynamic conflict-free coloring of intervals. Suppose that on any sequence of $n > 0$ insertions,* ALG *uses at most $c(n)$ colors and $r(n)$ recolorings per insertion, where $r(n) > 0$. Then $r(n) \geqslant n^{1/(c(n)+2)} - 2$.*

References

[1] A. Bar-Noy, P. Cheilaris, S. Olonetsky, and S. Smorodinsky. Online conflict-free colouring for hypergraphs. *Combinatorics, Probability & Computing*, 19(4):493–516, 2010.

[2] A. Bärtschi and F. Grandoni. On conflict-free multi-coloring. In *WADS*, pages 103–114. 2015.

[3] P. Cheilaris, L. Gargano, A. A. Rescigno, and S. Smorodinsky. Strong conflict-free coloring for intervals. *Algorithmica*, 70(4):732–749, 2014.

[4] K. Chen. How to play a coloring game against a color-blind adversary. In *SoCG*, pages 44–51, 2006.

[5] M. de Berg, T. Leijsen, A. Markovic, A. van Renssen, M. Roeloffzen, and G. Woeginger. Dynamic and kinetic conflict-free coloring of intervals with respect to points. *CoRR*, abs/1701.03388, 2017.

[6] G. Even, Z. Lotker, D. Ron, and S. Smorodinsky. Conflict-free colorings of simple geometric regions with applications to frequency assignment in cellular networks. *SIAM J. Comput.*, 33(1):94–136, 2003.

[7] A. Fiat, M. Levy, J. Matousek, E. Mossel, J. Pach, M. Sharir, S. Smorodinsky, U. Wagner, and E. Welzl. Online conflict-free coloring for intervals. In *SODA*, pages 545–554, 2005.

[8] S. Har-Peled and S. Smorodinsky. Conflict-free coloring of points and simple regions in the plane. *Discr. Comput. Geom.*, 34(1):47–70, 2005.

[9] E. Horev, R. Krakovski, and S. Smorodinsky. Conflict-free coloring made stronger. In *SWAT*, pages 105–117, 2010.

[10] S. Smorodinsky. *Combinatorial Problems in Computational Geometry*. PhD thesis, Tel-Aviv University, 2003.

[11] S. Smorodinsky. Conflict-free coloring and its applications. In *Geometry Intuitive, Discrete, and Convex: A Tribute to László Fejes Tóth*, pages 331–389. Springer Berlin Heidelberg, 2013.

Fine-grained complexity of coloring unit disks and balls

Csaba Biró*, Édouard Bonnet†, Dániel Marx†, Tillmann Miltzow†, Paweł Rzążewski†‡

Abstract

We investigate the possible complexity of an algorithm deciding the ℓ-colorability of an intersection graph of unit disks. We exhibit a smooth increase of complexity as the number ℓ of colors increases: If we restrict the number of colors to $\ell = \Theta(n^\alpha)$ for some $0 \leq \alpha \leq 1$, then the problem of coloring the intersection graph of n unit disks with ℓ colors

- can be solved in time $\exp\left(O(\sqrt{n\ell}\log n)\right)$, and
- cannot be solved in time $\exp\left(o(\sqrt{n\ell})\right)$, unless the ETH fails.

More generally, we consider the problem of coloring d-dimensional unit balls in the Euclidean space and obtain analogous results showing that the problem

- can be solved in time $\exp\left(O(n^{1-1/d}\ell^{1/d}\log n)\right)$, and
- cannot be solved in time $\exp\left(O(n^{1-1/d-\epsilon}\ell^{1/d})\right)$ for any $\epsilon > 0$, unless the ETH fails.

1 Introduction

On planar graphs, many classic algorithmic problems enjoy a certain "square root phenomenon" and can be solved significantly faster than what is known to be possible on general graphs: for example, Independent Set, 3-Coloring, Hamiltonian Cycle, Dominating Set can be solved in time $2^{O(\sqrt{n})}$ on an n-vertex planar graph, while no $2^{o(n)}$ algorithms exist for general graphs, assuming the Exponential Time Hypothesis (ETH) of Impagliazzo, Paturi, and Zane [2]. The square root in the exponent seems to be best possible for planar graphs: assuming the ETH, the running time for these problems cannot be improved to $2^{o(\sqrt{n})}$.

In some cases, a similar speedup can be obtained for 2-dimensional geometric problems, for example, there are $2^{O(\sqrt{n}\log n)}$ time algorithms for Independent Set on unit disk graphs or for TSP on 2-dimensional point sets [5, 1]. More generally, for d-dimensional geometric problems, running times of the from $2^{O(n^{1-1/d})}$ or $n^{O(k^{1-1/d})}$ appear naturally, and Marx and Sidiropoulos [4] showed that, assuming the ETH, this form of running time is essentially best possible for some problems.

In this paper, we explore whether such a speedup is possible for geometric coloring problems. Let us consider now the problem of coloring the intersection graph of a set of unit disks in the 2-dimensional plane, that is, assigning a color to each disk such that if two disks intersect, then they receive different colors. For a constant number of colors, geometric objects can behave similarly to planar graphs: 3-Coloring can be solved in time $2^{O(\sqrt{n})}$ on the intersection graph of n unit disks in the plane and, assuming the ETH, there is no such algorithm with running time $2^{o(\sqrt{n})}$. However, while every planar graph is 4-colorable, unit disks graphs can contain arbitrary large cliques, and hence the ℓ-colorability is a meaningful question for larger, non-constant, values of ℓ as well. We show that if the number ℓ of colors is part of the input and can be up to $\Theta(n)$, then, surprisingly, no speedup is possible: Coloring the intersection graph of n unit disks with ℓ colors cannot be solved in time $2^{o(n)}$, assuming the ETH. What happens between these two extremes of constant number of colors and $\Theta(n)$ colors? Our main 2-dimensional result exhibits a smooth increase of complexity as the number ℓ of colors increases.

Theorem 1 *For any fixed $0 \leq \alpha \leq 1$, the problem of coloring the intersection graph of n unit disks with $\ell = \Theta(n^\alpha)$ colors*

- *can be solved in time $2^{O(n^{\frac{1+\alpha}{2}}\log n)} = 2^{O(\sqrt{n\ell}\log n)}$, and*
- *cannot be solved in time $2^{o(n^{\frac{1+\alpha}{2}})} = 2^{o(\sqrt{n\ell})}$, unless the ETH fails.*

The proof is not very specific to disks and can be easily adapted to, say, axis-parallel unit squares or other fat objects. However, it seems that the requirement of fatness is essential for this type of complexity behavior as, for example, the coloring of the intersection graphs of line segments (of arbitrary lengths) does not admit

*Department of Mathematics, University of Louisville

†Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI). Supported by the ERC grant PARAMTIGHT: "Parameterized complexity and the search for tight complexity results", no. 280152.

‡Faculty of Mathematics and Information Science, Warsaw University of Technology

any speedup compared to the $2^{O(n)}$ algorithm, even for a constant number of colors.

Theorem 2 *There is no $2^{o(n)}$ time algorithm for* 6-Coloring *the intersection graph of line segments in the plane, unless the ETH fails.*

How does the complexity change if we look at the generalization of the coloring problem into higher dimensions? It is known for some problems that if we generalize the problem from two dimensions to d dimensions, then the square root in the exponent of the running time changes to a $1-1/d$ power, which makes the running time closer and closer to the running time of the brute force as d increases. For the ℓ-coloring problem, the correct exponent seems to be $n^{1-1/d}$ times $\ell^{1/d}$. That is, as d increases, the running time becomes less and less sensitive to the number of colors and approaches $2^{O(n)}$, even for constant number of colors.

Theorem 3 *For any fixed $0 \leq \alpha \leq 1$ and dimension $d \geq 2$, the problem of coloring the intersection graph of n unit balls in the d-dimensional Euclidean space with $\ell = \Theta(n^\alpha)$ colors*

- *can be solved in time $2^{O\left(n^{\frac{d-1+\alpha}{d}} \log n\right)} = 2^{O\left(n^{1-1/d}\ell^{1/d} \log n\right)}$, and*
- *cannot be solved in time $2^{n^{\frac{d-1+\alpha}{d}-\epsilon}}$ for any $\epsilon > 0$, unless the ETH fails.*

The upper bounds of Theorems 1 and 3 follow fairly easily using standard techniques. Clearly, the problem of coloring unit d-balls with ℓ colors makes sense only if every point of the space is contained in at most ℓ balls: otherwise we would immediately know that there is no ℓ-coloring. On the other hand, if every point is contained in at most ℓ of the n balls, then it is known that there is a balanced separator of size $O(n^{1-1/d}\ell^{1/d})$ [5]. By finding such a separator and trying every possible coloring on the disks of the separator, we can branch into $\ell^{O(n^{1-1/d}\ell^{1/d})}$ smaller instances. This recursive procedure has the running time as claimed.

2 Auxiliary problems

We start with introducing two auxiliary problems, which will serve as middle steps in the hardness reduction. For a fixed dimension d and $i \in [d]$, we denote by e_i the d-dimensional vector, whose i-th coordinate is equal to 1 and all remaining coordinates are equal to 0. For two positive integers g, d, we denote by $R[g,d]$ the *d-dimensional grid*, i.e., a graph whose vertices are all vectors from $[g]^d$, and two vertices are adjacent if they differ on exactly one coordinate, and exactly by one (on that coordinate). In other words, a and a' are adjacent if $a = a' \pm e_i$ for some $i \in [d]$. We will often refer to vertices of a grid as *cells*.

Problem: d-grid 3-Sat
Input: A d-dimensional grid $G = R[g,d]$, $k \in \mathbb{N}_+$, a function $\zeta : v \in V(G) \mapsto \{v_1, v_2, \ldots, v_k\}$ mapping each cell v to k fresh boolean variables, and a set $\mathcal{C}$ of constraints of two kinds:

clause constraints: for a cell v, a set $\mathcal{C}(v)$ of pairwise variable-disjoint disjunctions of at most 3 literals on $\zeta(v)$;
equality constraints: for adjacent cells v and w, a set $\mathcal{C}(v,w)$ of pairwise variable-disjoint constraints of the form $v_i = w_j$ (with $i, j \in [k]$).

Question: Is there an assignment of the variables such that all constraints are satisfied?

The size of the instance $I = (G, k, \zeta, \mathcal{C})$ of d-grid 3-Sat is the total number of variables, i.e., $g^d k$.

Problem: Partial d-grid Coloring
Input: An induced subgraph G of $R[g,d]$, $\ell \in \mathbb{N}_+$, and a function $\rho : v \in V(G) \mapsto \{p_1^v, p_2^v, \ldots, p_\ell^v\} \in ([\ell]^d)^\ell$ mapping each cell v to a set of ℓ points in $[\ell]^d$.
Question: Is there an ℓ-coloring of all the points such that:

- two points in the same cell get different colors;
- if v and w are adjacent in G, say, $w = v + e_i$ (for some $i \in [d]$), and $p \in \rho(v)$ and $q \in \rho(w)$ receive the same color, then $p[i] \leqslant q[i]$ where $a[i] := a \cdot e_i$ is the i-th coordinate of a?

Here the size of the instance is the total number of points, i.e., $|V(F)|\ell \leq g^d \ell$.

3 Hardness of coloring unit disks

First, by a reduction from 3-Sat, we show that 2-grid 3-Sat with total size n and k variables per cell cannot be solved in time $2^{o(\sqrt{nk})}$, unless the ETH fails. The main result of this section is the following theorem.

Theorem 4 *For any $0 \leqslant \alpha \leqslant 1$, there is no $2^{o(\sqrt{n\ell})}$ algorithm solving* Partial 2-grid Coloring *on a total of n points and $\ell = \Theta(n^\alpha)$ points in each cell (that is n/ℓ cells), unless the ETH fails.*

Proof (sketch). We present a reduction from 2-GRID 3-SAT to PARTIAL 2-GRID COLORING. Let $I = (G, k, \zeta, \mathcal{C})$ be an instance of 2-GRID 3-SAT, where $G = R[g, 2]$ and each cell contains k variables. We construct an equivalent instance $J = (F, \ell, \rho)$ of PARTIAL 2-GRID COLORING with $|V(F)| = \Theta(|V(G)|) = \Theta(g^2)$ and $\ell := 4k$ points per cell, where F is an induced subgraph of $R[g', 2]$ with $g' = \Theta(g)$. Let us present the key ideas of the construction.

Standard cells. A *standard cell* is a cell where the points $p_1, \ldots, p_\ell$ are on the main diagonal, that is $p_i = (i, i)$ for every $i \in [\ell]$ (see cells in Fig. 1).

Reference coloring. In the construction we will choose one standard cell $\bar{R}$, whose coloring will be referred to as the *reference coloring*. Now, by the color $i \in [\ell]$, we mean the color of the point p_i in $\bar{R}$.

Variable-assignment cells. For each cell $v = (i, j) \in V(G)$, we introduce in F a standard cell $A(v)$), called the *variable-assignment cell*. The cell $A(v)$ is responsible for encoding the truth assignment of variables in $\zeta(v)$. If $i+j$ is even, then the cell $A(v)$ is also called *even*. Otherwise $A(v)$ is *odd*.

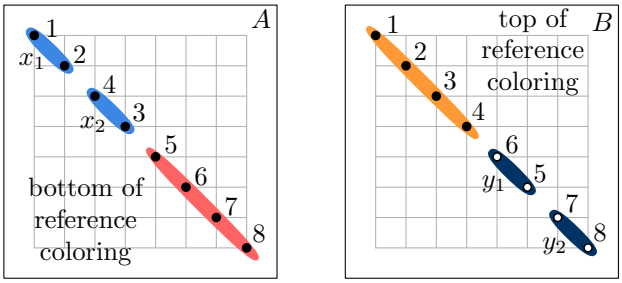


Figure 1: Even (left) and odd (right) variable-assignment cells.

In our construction, we make sure that each variable-assignment cell receives one of the *standard colorings*. If $A(v)$ is even, the coloring φ of $A(v)$ is standard if $\{\varphi(p_{2i-1}), \varphi(p_{2i})\} = \{2i-1, 2i\}$ for $i \in [k]$ and $\varphi(p_i) = i$ for $i \in [4k] \setminus [2k]$. If the cell $A(v)$ is odd, its standard colorings φ are the ones with $\varphi(p_i) = i$ for $i \in [2k]$ and $\{\varphi(p_{2i-1}), \varphi(p_{2i})\} = \{2i-1, 2i\}$ for $i \in [2k] \setminus [k]$. The choice of the particular standard coloring for the points in $A(v)$ defines the actual assignment of variables in $\zeta(v)$. If $A(v)$ is even, then for each $i \in [k]$, we interpret the coloring in the following way:

$p_{2i-1} \mapsto 2i-1$, $p_{2i} \mapsto 2i$ as setting v_i to true;
$p_{2i-1} \mapsto 2i$, $p_{2i} \mapsto 2i-1$ as setting v_i to false.

If $A(v)$ is odd, for each $i \in [k]$, we interpret it in that way:

$p_{2k+2i-1} \mapsto 2i-1$, $p_{2k+2i} \mapsto 2i$ as setting v_i to true;
$p_{2k+2i-1} \mapsto 2i$, $p_{2k+2i} \mapsto 2i-1$ as setting v_i to false.

Local reference cells. For each inner face of G (see Fig. 2), we introduce a new standard cell, called a *local reference cell*. Moreover, we set the reference $\bar{R}$ to be uppermost-leftmost local reference cell. In the construction, we will ensure that the coloring of each local reference cell is exactly the same, i.e., is exactly the reference coloring.

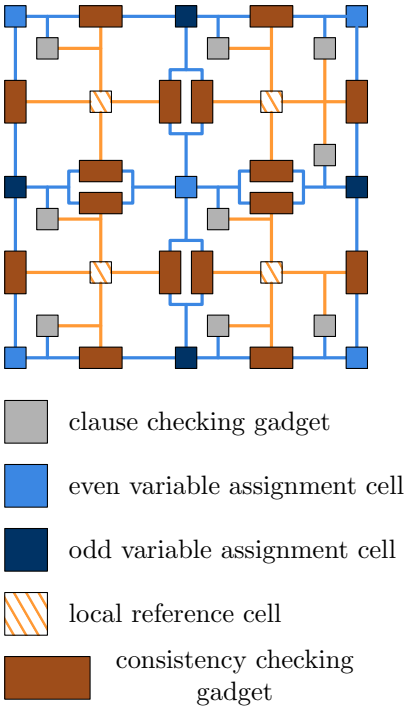


Figure 2: High-level illustration of J.

Overview of the construction. Figure 2 presents the arrangement of the cells in F. For each variable-assignment cell $A(v)$, we introduce a *clause-checking gadget*, which is responsible for ensuring that all clauses in $\mathcal{C}(v)$ are satisfied. This gadget requires an access to the reference coloring, which we can attain from the local reference cells (we can choose any of the local reference cells close to $A(v)$). For each edge vw of G, we introduce a *consistency gadget*. In fact, for inner edges of G (i.e., the ones not incident with the outer face) we introduce two consistency gadgets, one for each face incident with vw. This gadget is responsible for ensuring the consistency on three different levels:

- to force satisfying all equality constraints $\mathcal{C}(v, w)$,
- to ensure that each of $A(v)$ and $A(w)$ receives one of the standard colorings,
- to ensure that the local reference cell contains exactly the reference coloring.

This gadget also requires access to the reference coloring, so we join it with the appropriate local reference cell (see Fig. 2).

Now, we observe that the total number of points in F is $n = O(g^2\ell) = O(n')$, where $n' = g^2k$ is the total size of I. Thus, the existence of an algorithm solving J in time $2^{o(\sqrt{n\ell})}$ could be used to solve I in time $2^{o(\sqrt{n'k})}$, which, in turn, contradicts the ETH. ■

Now, to prove the lower bound in Theorem 1, we need to show a reduction from Partial 2-grid Coloring to the problem of coloring unit disk graphs. This reduction uses a well-known approach [3, Theorems 1 and 3], presented on Fig. 3.

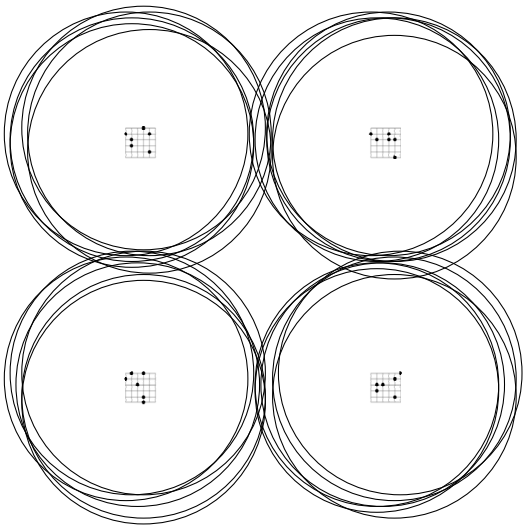


Figure 3: Reduction from Partial 2-grid Coloring to coloring unit disks.

4 Hardness of coloring unit d-balls for $d \geq 3$

The d-dimensional lower bound of Theorem 3 goes along the same lines, but we first prove a lower bound for d-grid 3-Sat. Based on earlier results by Marx and Sidiropoulos [4], we prove an almost tight lower bound for this d-dimensional 3-SAT by embedding a 3-SAT instance with roughly $g^{d-1}k$ variables and clauses into the d-dimensional $g \times \cdots \times g$-grid $R[g, d]$. Then the reduction from this problem to coloring unit balls in d-dimensional space is very similar to the 2-dimensional case.

5 Hardness of coloring segments

Finally, we show that fatness is necessary to obtain subexponential-time algorithm for coloring.

Proof of Theorem 2 (sketch). We reduce from 3-coloring of graphs with maximum degree at most 4, which cannot be solved in time $2^{o(n)}$, assuming the ETH. Let G be a graph with n vertices $v_1, v_2, \ldots, v_n$. For each vertex v_i we introduce two segments: a horizontal one, called x_i, and a vertical one, called y_i, so that they form a half of a $n \times n$ grid (see Figure 4). Using appropriate gadgets we ensure that each x_i can only receive colors $\{1, 2, 3\}$, while each y_i can only receive colors $\{4, 5, 6\}$. Each color $c \in \{1, 2, 3\}$

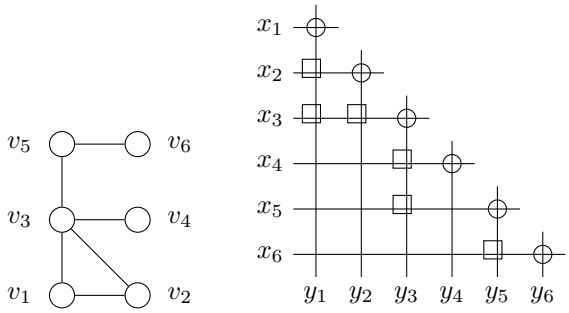


Figure 4: High-level idea of the construction.

will be identified with the color $c + 3$. We want to ensure that in any feasible 6-coloring f of G' we have $f(x_i) + 3 = f(y_i)$ for all $i \in [n]$, and $f(x_i) + 3 \neq f(y_i)$ for all $i > j$ such that v_iv_j is an edge of G. This is achieved by using constant-size *equality gadgets* and *inequality gadgets*. At the crossing point of x_i and y_i, we put an equality gadget (represented by a circle on Fig. 4). Moreover, for each edge v_iv_j of G, we put an inequality gadget at the crossing point of x_i and y_j, $i > j$ (represented by a square on Fig. 4). ■

References

[1] J. Alber and J. Fiala. Geometric separation and exact solutions for the parameterized independent set problem on disk graphs. *J. Algorithms*, 52(2):134 151, 2004.

[2] R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512 530, 2001.

[3] D. Marx. Efficient approximation schemes for geometric problems? In *ESA 2005 Proc.*, pages 448 459, 2005.

[4] D. Marx and A. Sidiropoulos. The limited blessing of low dimensionality: When 1-1/d is the best possible exponent for d-dimensional geometric problems. SOCG 2014 Proc., pages 67:67 67:76, New York, NY, USA, 2014. ACM.

[5] W. D. Smith and N. C. Wormald. Geometric separator theorems and applications. FOCS 1998 Proc., pages 232 243, Washington, DC, USA, 1998. IEEE Computer Society.

Coloring curves that cross a fixed curve*

Alexandre Rok† Bartosz Walczak‡

Abstract

We prove that for every integer $t \geqslant 1$, the class of intersection graphs of curves in the plane each of which crosses a fixed curve in at least one and at most t points is χ-bounded. This is essentially the strongest χ-boundedness result one can get for this kind of graph classes. As a corollary, we prove that for any fixed integers $k \geqslant 2$ and $t \geqslant 1$, every k-quasi-planar topological graph on n vertices with any two edges crossing at most t times has $O(n \log n)$ edges.

1 Introduction

Overview A class of graphs is *χ-bounded* if every graph G in the class satisfies $\chi(G) \leqslant f(\omega(G))$ for some function $f\colon \mathbb{N} \to \mathbb{N}$, where $\chi(G)$ and $\omega(G)$ denote the chromatic number and the clique number (the maximum size of a clique) of G, respectively. A *curve* is a homeomorphic image of the real interval $[0, 1]$ in the plane. The *intersection graph* of a family $\mathcal{F}$ of curves has $\mathcal{F}$ as the set of vertices and the intersecting pairs of curves in $\mathcal{F}$ as the set of edges. Intersection graphs of curves are known as *string graphs*. Although the class of all string graphs is not χ-bounded [16, 17], all known constructions of string graphs with small clique number and large chromatic number require a lot of freedom in placing curves around in the plane.

What restrictions on placement of curves lead to χ-bounded classes of intersection graphs? McGuinness [13, 14] proposed studying families of curves that cross a fixed curve *exactly once*. This initiated a series of results culminating in the proof that the class of intersection graphs of such families is indeed χ-bounded [18]. We prove an essentially farthest possible generalization of this result, allowing curves to cross the fixed curve *at least once and at most t times*, for any bound t.

Theorem 1 *For any $t \geqslant 1$ and any fixed curve c_0, the class of intersection graphs of curves each crossing c_0 in at least one and at most t points is χ-bounded.*

By contrast, the class of intersection graphs of curves each crossing a fixed curve *at least once* is equal to the class of all string graphs and therefore is not χ-bounded. Additional motivation for Theorem 1 comes from its application to bounding the number of edges in k-quasi-planar graphs (see the next page).

Context Chromatic number of intersection graphs of geometric objects has been investigated since the 1960s. Asplund and Grünbaum [3] proved that intersection graphs of axis-parallel rectangles in the plane satisfy $\chi = O(\omega^2)$ and conjectured that the class of intersection graphs of axis-parallel boxes in $\mathbb{R}^d$ is χ-bounded also for all $d \geqslant 3$. However, a surprising construction due to Burling [5] showed that there are triangle-free intersection graphs of axis-parallel boxes in $\mathbb{R}^3$ with arbitrarily large chromatic number. Another classical example of a χ-bounded class of geometric intersection graphs is provided by circle graphs (intersection graphs of chords of a fixed circle) [9].

McGuinness [13, 14] proposed investigating the problem when much more general geometric shapes are allowed but the way how they are arranged in the plane is restricted. In [13], he proved that the class of intersection graphs of L-shapes crossing a fixed horizontal line is χ-bounded. Families of L-shapes in the plane are *simple*, which means that any two members of the family intersect in at most one point. McGuinness [14] also showed that triangle-free intersection graphs of simple families of curves each crossing a fixed line in exactly one point have bounded chromatic number. Further progress in this direction was made by Suk [19], who proved that simple families of x-monotone curves crossing a fixed vertical line give rise to a χ-bounded class of intersection graphs, and by Lasoń et al. [12], who reached the same conclusion without assuming that the curves are x-monotone. Finally, in [18], we proved that the class of intersection graphs of curves each crossing a fixed line in exactly one point is χ-bounded. These results remain valid if the fixed straight line is replaced by a fixed curve [20].

The class of string graphs is not χ-bounded. Pawlik et al. [16, 17] presented a construction of triangle-free intersection graphs of segments (or geometric shapes of various other kinds) with chromatic number growing as fast as $\Theta(\log \log n)$ with the number of vertices n. It was further generalized to a construction of string graphs with clique number ω and chromatic number $\Theta_\omega((\log \log n)^{\omega-1})$ [11]. The best known upper bound on the chromatic number of string graphs in terms of the number of vertices is $(\log n)^{O(\log \omega)}$, due to Fox

*Full version is available at arXiv:1512.06112.

†Department of Mathematics, Ben-Gurion University of the Negev, Be'er Sheva, Israel, rok@math.bgu.ac.il; partially supported by Israel Science Foundation grant 1136/12.

‡Department of Theoretical Computer Science, Faculty of Mathematics and Computer Science, Jagiellonian University, Kraków, Poland, walczak@tcs.uj.edu.pl; partially supported by National Science Center of Poland grant 2015/17/D/ST1/00585.

and Pach [8]. For intersection graphs of segments (x-monotone curves), an upper bound of the form $\chi = O_\omega(\log n)$ follows from the above-mentioned result in [19] ([18]) via recursive halving. Upper bounds of the form $\chi = O_\omega((\log\log n)^{f(\omega)})$ (for some $f\colon \mathbb{N} \to \mathbb{N}$) are known for very special classes of string graphs: rectangle overlap graphs [10, 11] and subtree overlap graphs [11]. The former still allow the triangle-free construction with $\chi = \Theta(\log\log n)$ and the latter the construction with $\chi = \Theta_\omega((\log\log n)^{\omega-1})$.

Quasi-planarity A *topological graph* is a graph with a fixed curvilinear drawing in the plane. For $k \geqslant 2$, a *k-quasi-planar graph* is a topological graph with no k pairwise crossing edges. It is conjectured that k-quasi-planar graphs with n vertices have $O_k(n)$ edges [4, Problem 1 in Section 9.6]. For $k = 2$, this asserts a well-known property of planar graphs. The conjecture is also verified for $k = 3$ [2, 15] and $k = 4$ [1], but it remains open for $k \geqslant 5$. Best known upper bounds on the number of edges in a k-quasi-planar graph are $n(\log n)^{O(\log k)}$ in general [7, 8], $O_k(n \log n)$ for the case of x-monotone edges [21], $O_k(n \log n)$ for the case that any two edges intersect at most once [20], and $2^{\alpha(n)^\nu} n \log n$ for the case that any two edges intersect in at most t points, where α is the inverse Ackermann function and ν depends on k and t [20]. We apply Theorem 1 to improve the last bound to $O_{k,t}(n \log n)$, following verbatim the proof in [20] for the case $t = 1$.

Theorem 2 *Every k-quasi-planar topological graph G on n vertices such that any two edges of G intersect in at most t points has at most $O_{k,t}(n \log n)$ edges.*

2 Proof sketch of Theorem 1

Setup Graph-theoretic terms (like chromatic number, clique, etc.) applied to a family of curves $\mathcal{F}$ have the same meaning as applied to the intersection graph of $\mathcal{F}$. From now on, without significant loss of generality, we make the following implicit assumption: any two curves that we consider intersect in finitely many points, and each of their intersection points is a proper crossing (however, a curve c may have an endpoint on another curve if this is required by the definition of c).

Theorem 1 (rephrased) *For every $t \in \mathbb{N}$, there is a non-decreasing function $f_t\colon \mathbb{N} \to \mathbb{N}$ with the following property: for any fixed curve c_0, every family $\mathcal{F}$ of curves each intersecting c_0 in at least one and at most t points satisfies $\chi(\mathcal{F}) \leqslant f_t(\omega(\mathcal{F}))$.*

Initial reduction We fix a horizontal line in the plane and call it the *baseline*. The upper halfplane bounded by the baseline is denoted by H^+. A 1-*curve* is a curve in H^+ that has one endpoint on the baseline and does not intersect the baseline in any other point. Intersection graphs of 1-*curves* are known as *outerstring graphs*. The starting point of the proof of Theorem 1 is the following result, due to the authors.

Theorem 3 [18] *There is a non-decreasing function $f_0\colon \mathbb{N} \to \mathbb{N}$ such that every family $\mathcal{F}$ of 1-curves satisfies $\chi(\mathcal{F}) \leqslant f_0(\omega(\mathcal{F}))$.*

An *even-curve* is a curve that has both endpoints above the baseline and intersects the baseline in at least two points (this is an even number, by the proper crossing assumption). For $t \in \mathbb{N}$, a $2t$-*curve* is an even-curve that intersects the baseline in exactly $2t$ points. The *basepoint* of a 1-curve s is the endpoint of s on the baseline. A *basepoint* of an even-curve c is an intersection point of c with the baseline. Every even-curve c determines two 1-curves—the two parts of c from an endpoint to the closest basepoint. They are called the 1-*curves of* c and denoted by $L(c)$ and $R(c)$ so that the basepoint of $L(c)$ lies to the left of the basepoint of $R(c)$ on the baseline. A family $\mathcal{F}$ of even-curves is an *LR-family* if every intersection between two curves $c_1, c_2 \in \mathcal{F}$ is an intersection between $L(c_1)$ and $R(c_2)$ or between $L(c_2)$ and $R(c_1)$. The main effort in this work goes to proving the following statement.

Theorem 4 *There is a non-decreasing function $f\colon \mathbb{N} \to \mathbb{N}$ such that every LR-family $\mathcal{F}$ of even-curves satisfies $\chi(\mathcal{F}) \leqslant f(\omega(\mathcal{F}))$.*

Lemma 5 *For every $t \in \mathbb{N}$, there is a non-decreasing function $f_t\colon \mathbb{N} \to \mathbb{N}$ such that every family $\mathcal{F}$ of $2t$-curves no two of which intersect on or below the baseline satisfies $\chi(\mathcal{F}) \leqslant f_t(\omega(\mathcal{F}))$.*

Theorem 1 is reduced to Lemma 5 as follows. First, we surround c_0 very closely by a closed curve γ intersecting every curve in the family in exactly $2t$ points (winding if necessary). Then, we invert the plane "inside out" with respect to γ and unfold γ to a horizontal line. Lemma 5 is proved by induction on t with Theorem 3 used for the base case and Theorem 4 used for the induction step. It remains to prove Theorem 4.

In an LR-family of even-curves $\mathcal{F}$, only the 1-curves $L(c)$ and $R(c)$ of any curve $c \in \mathcal{F}$ participate in intersections with other curves in $\mathcal{F}$, and the part of c connecting $L(c)$ and $R(c)$ remains disjoint from all other curves in $\mathcal{F}$. It turns out that these "middle" parts connecting the two 1-curves of even-curves in $\mathcal{F}$ are essential for Theorem 4 to hold. To state this formally, we define a *double-curve* as a set $X \subseteq H^+$ that is a union of two disjoint 1-curves, denoted by $L(X)$ and $R(X)$ so that the basepoint of $L(X)$ lies to the left of the basepoint of $R(X)$, and we call a family $\mathcal{X}$ of double-curves an *LR-family* if every intersection between two double-curves $X_1, X_2 \in \mathcal{X}$ is an intersection between $L(X_1)$ and $R(X_2)$ or between $L(X_2)$ and $R(X_1)$.

Theorem 6 *For every $\zeta \in \mathbb{N}$, there is a triangle-free LR-family of double-curves $\mathcal{X}$ such that $\chi(\mathcal{X}) \geqslant \zeta$.*

The proof of Theorem 6 is an easy adaptation of the construction from [16, 17].

More notation and terminology Let $\prec$ denote the left-to-right order of points on the baseline ($p_1 \prec p_2$ means that p_1 is to the left of p_2). We also use the notation $\prec$ for curves intersecting the baseline ($c_1 \prec c_2$ means that every basepoint of c_1 is to the left of every basepoint of c_2) and for families of such curves ($\mathcal{C}_1 \prec \mathcal{C}_2$ means that $c_1 \prec c_2$ for any $c_1 \in \mathcal{C}_1$ and $c_2 \in \mathcal{C}_2$). For a family $\mathcal{C}$ of curves intersecting the baseline (even-curves or 1-curves) and two 1-curves x and y, let $\mathcal{C}(x,y) = \{c \in \mathcal{C} \colon x \prec c \prec y \text{ or } y \prec c \prec x\}$. For a family $\mathcal{C}$ of curves intersecting the baseline and a segment I on the baseline, let $\mathcal{C}(I)$ denote the family of curves in $\mathcal{C}$ with all basepoints on I.

For an even-curve c, let $M(c)$ denote the subcurve of c connecting the basepoints of $L(c)$ and $R(c)$, and let $I(c)$ denote the segment on the baseline connecting the basepoints of $L(c)$ and $R(c)$. For a family $\mathcal{F}$ of even-curves, let $L(\mathcal{F}) = \{L(c) \colon c \in \mathcal{F}\}$, $R(\mathcal{F}) = \{R(c) \colon c \in \mathcal{F}\}$, and $I(\mathcal{F})$ denote the minimal segment on the baseline that contains $I(c)$ for every $c \in \mathcal{F}$.

A *cap-curve* is a curve in H^+ that has both endpoints on the baseline and does not intersect the baseline in any other point. For a cap-curve γ, the set $H^+ \setminus \gamma$ has two connected components: one bounded, denoted by $\operatorname{int}\gamma$, and one unbounded, denoted by $\operatorname{ext}\gamma$.

Proof sketch of Theorem 4 First, we reduce Theorem 4 to the following special case of it.

Lemma 7 *There is a non-decreasing function $f \colon \mathbb{N} \to \mathbb{N}$ such that every LR-family $\mathcal{F}$ of 2-curves satisfies $\chi(\mathcal{F}) \leqslant f(\omega(\mathcal{F}))$.*

Lemma 8 *For every LR-family of even-curves $\mathcal{F}$, if $\mathcal{F}^\star$ is the family of curves $c \in \mathcal{F}$ such that $L(c)$ and $R(c)$ lie in two distinct connected components of the union of all 1-curves in $L(\mathcal{F}) \cup R(\mathcal{F})$, then $\chi(\mathcal{F}^\star) \leqslant 4$.*

We prove Lemma 8 by showing that the intersection graph of $\mathcal{F}^\star$ is planar. Then, to prove Theorem 4 from Lemma 7, we show that $I(c_1)$ and $I(c_2)$ are nested or disjoint for any $c_1, c_2 \in \mathcal{F} \setminus \mathcal{F}^\star$, which easily implies that $\mathcal{F} \setminus \mathcal{F}^\star$ is equivalent to an *LR*-family of 2-curves.

For $\xi \in \mathbb{N}$, a *ξ-family* is an *LR*-family of 2-curves $\mathcal{F}$ with the following property: for every 2-curve $c \in \mathcal{F}$, the family of 2-curves in $\mathcal{F} \setminus \{c\}$ that intersect c has chromatic number at most ξ. We use induction on $\omega(\mathcal{F})$ to reduce Lemma 7 to the following statement.

Lemma 9 *For any $\xi, k \in \mathbb{N}$, there is $\zeta \in \mathbb{N}$ such that every ξ-family $\mathcal{F}$ with $\omega(\mathcal{F}) \leqslant k$ satisfies $\chi(\mathcal{F}) \leqslant \zeta$.*

Lemma 10 *For every $\xi \in \mathbb{N}$, every ξ-family $\mathcal{F}$ with $\bigcap_{c \in \mathcal{F}} I(c) \neq \emptyset$ satisfies $\chi(\mathcal{F}) \leqslant 4\xi + 4$.*

Lemma 10 is proved as in [20, Lemma 19], using the following elementary lemma due to McGuinness.

Lemma 11 [13, Lemma 2.1] *Let G be a graph, $\prec$ be a total order on the vertices of G, and $\alpha, \beta \in \mathbb{N}$. If $\chi(G) > (2\beta+2)\alpha$, then G has an induced subgraph H such that $\chi(H) > \alpha$ and $\chi(G(u,v)) > \beta$ for every edge uv of H, where $G(u,v)$ denotes the subgraph of G induced on the vertices strictly between u and v in $\prec$.*

Lemma 11 easily implies that every family of 2-curves $\mathcal{F}$ with $\chi(\mathcal{F}) > (2\beta+2)^2\alpha$ contains a subfamily $\mathcal{H}$ with $\chi(\mathcal{H}) > \alpha$ such that $\chi(\mathcal{F}(L(c_1), L(c_2))) > \beta$ and $\chi(\mathcal{F}(R(c_1), R(c_2))) > \beta$ for any two intersecting 2-curves $c_1, c_2 \in \mathcal{H}$. This is considerably strengthened by the following lemma. Its proof extends the idea used in [13] for the proof of Lemma 11.

Lemma 12 *For every $\xi \in \mathbb{N}$, there is a function $f \colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ with the following property: for any $\alpha, \beta \in \mathbb{N}$ and every ξ-family $\mathcal{F}$, if $\chi(\mathcal{F}) > f(\alpha, \beta)$, then there is a subfamily $\mathcal{H} \subseteq \mathcal{F}$ such that $\chi(\mathcal{H}) > \alpha$ and $\chi(\mathcal{F}(x,y)) > \beta$ for any two intersecting 1-curves $x \in R(\mathcal{H})$ and $y \in L(\mathcal{H})$.*

It is proved in [18] that for every family of 1-curves $\mathcal{S}$, there are a cap-curve γ and a subfamily $\mathcal{U} \subseteq \mathcal{S}$ with $\chi(\mathcal{U}) \geqslant \frac{1}{2}\chi(\mathcal{S})$ such that every 1-curve in $\mathcal{U}$ lies in $\operatorname{int}\gamma$ and intersects some 1-curve in $\mathcal{S}$ that intersects $\operatorname{ext}\gamma$. The proof follows an idea from [9], defining $\mathcal{U}$ as one of the sets of 1-curves at a fixed distance from an appropriately chosen 1-curve in the intersection graph of $\mathcal{S}$. However, this method fails to imply an analogous statement for 2-curves. We need a more powerful tool (very recent) due to Chudnovsky, Scott, and Seymour.

Theorem 13 [6, Theorem 1.8] *There is a function $f \colon \mathbb{N} \to \mathbb{N}$ with the following property: for every $\alpha \in \mathbb{N}$, every string graph G with $\chi(G) > f(\alpha)$ contains a vertex v such that $\chi(G_v^2) > \alpha$, where G_v^2 denotes the subgraph of G induced on the vertices within distance at most 2 from v.*

This and Lemma 10 easily yield the following.

Lemma 14 *For every $\xi \in \mathbb{N}$, there is a function $f \colon \mathbb{N} \to \mathbb{N}$ with the following property: for every $\alpha \in \mathbb{N}$ and every ξ-family $\mathcal{F}$ with $\chi(\mathcal{F}) > f(\alpha)$, there are a cap-curve γ and a subfamily $\mathcal{G} \subseteq \mathcal{F}$ with $\chi(\mathcal{G}) > \alpha$ such that every 2-curve $c \in \mathcal{G}$ has $L(c), R(c) \subseteq \operatorname{int}\gamma$ and intersects some 2-curve in $\mathcal{F}$ that intersects $\operatorname{ext}\gamma$.*

For $\xi \in \mathbb{N}$ and a function $h \colon \mathbb{N} \to \mathbb{N}$, a *$(\xi, h)$-family* is a ξ-family $\mathcal{F}$ with the following additional property: for every $\alpha \in \mathbb{N}$ and every subfamily $\mathcal{G} \subseteq \mathcal{F}$ with $\chi(\mathcal{G}) > h(\alpha)$, there is a subfamily $\mathcal{H} \subseteq \mathcal{G}$ with $\chi(\mathcal{H}) > \alpha$ such that every 2-curve in $\mathcal{F}$ with a basepoint on $I(\mathcal{H})$ has both basepoints on $I(\mathcal{G})$.

Lemma 15 *For any $\xi, k \in \mathbb{N}$ and any function $h \colon \mathbb{N} \to \mathbb{N}$, there is a constant $\zeta \in \mathbb{N}$ such that every (ξ, h)-family $\mathcal{F}$ with $\omega(\mathcal{F}) \leqslant k$ satisfies $\chi(\mathcal{F}) \leqslant \zeta$.*

Lemma 15 easily implies the next lemma. Then, the next lemma together with Lemma 15 easily imply Lemma 9, completing the proof of Theorem 4.

Lemma 16 *For any $\xi, k \in \mathbb{N}$, there is a function $f\colon \mathbb{N} \to \mathbb{N}$ such that for every $\alpha \in \mathbb{N}$, every ξ-family $\mathcal{F}$ with $\omega(\mathcal{F}) \leqslant k$ and $\chi(\mathcal{F}) > f(\alpha)$ contains a 2-curve c with $\chi(\mathcal{F}(I(c))) > \alpha$.*

The proof of Lemma 15 has similar structure to and borrows several ideas from the proof in [18].

For a family of 1-curves $\mathcal{S}$, an *$\mathcal{S}$-skeleton* is a pair $(\gamma,\mathcal{U})$ such that γ is a cap-curve, $\mathcal{U}$ is a family of pairwise disjoint 1-curves—subcurves of 1-curves in $\mathcal{S}$, and each 1-curve in $\mathcal{U}$ has one endpoint (other than the basepoint) on γ and all the remaining part in $\operatorname{int}\gamma$. A family of 2-curves $\mathcal{F}$ is *supported* by $\mathcal{S}$ if every 2-curve in $\mathcal{F}$ intersects some 1-curve in $\mathcal{S}$, and $\mathcal{F}$ is supported by an $\mathcal{S}$-skeleton $(\gamma,\mathcal{U})$ if every 2-curve $c \in \mathcal{F}$ satisfies $L(c), R(c) \subseteq \operatorname{int}\gamma$ and intersects some 1-curve in $\mathcal{U}$.

Lemma 17 *For every function $h\colon \mathbb{N} \to \mathbb{N}$, there is a function $f\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ with the following property: for any $\alpha, \beta \in \mathbb{N}$, every (ξ, h)-family $\mathcal{F}$ with $\chi(\mathcal{F}) > f(\alpha, \beta)$ contains one of the following structures:*

- *a subfamily $\mathcal{G} \subseteq \mathcal{F}$ with $\chi(\mathcal{G}) > \alpha$ supported by an $L(\mathcal{F})$-skeleton or an $R(\mathcal{F})$-skeleton,*
- *a subfamily $\mathcal{H} \subseteq \mathcal{F}$ with $\chi(\mathcal{H}) > \beta$ supported by a family of 1-curves $\mathcal{S}$ such that $\mathcal{S} \subseteq L(\mathcal{F})$ or $\mathcal{S} \subseteq R(\mathcal{F})$, and $s \prec \mathcal{H}$ or $\mathcal{H} \prec s$ for every $s \in \mathcal{S}$.*

Lemma 18 *For every function $h\colon \mathbb{N} \to \mathbb{N}$, there is a function $f\colon \mathbb{N} \to \mathbb{N}$ with the following property: for every $\alpha \in \mathbb{N}$, every (ξ, h)-family $\mathcal{F}$ with $\chi(\mathcal{F}) > f(\alpha)$ contains a subfamily $\mathcal{G} \subseteq \mathcal{F}$ with $\chi(\mathcal{G}) > \alpha$ supported by an $L(\mathcal{F})$-skeleton or an $R(\mathcal{F})$-skeleton.*

Lemma 19 *For every $\xi \in \mathbb{N}$ and every function $h\colon \mathbb{N} \to \mathbb{N}$, there is a function $f\colon \mathbb{N} \to \mathbb{N}$ with the following property: for every $n \in \mathbb{N}$ and every (ξ, h)-family $\mathcal{F}$ with $\chi(\mathcal{F}) > f(n)$, there are 2-curves $a_1, b_1, \ldots, a_n, b_n \in \mathcal{F}$ such that*

- *for $1 \leqslant i \leqslant n$, $R(a_i)$ and $L(b_i)$ intersect,*
- *for $2 \leqslant i \leqslant n$, the basepoints of $R(a_i)$ and $L(b_i)$ lie between the basepoints of $R(a_{i-1})$ and $L(b_{i-1})$,*
- *for $2 \leqslant i \leqslant n$, $L(a_i)$ intersects $R(a_1), \ldots, R(a_{i-1})$ or $R(b_i)$ intersects $L(b_1), \ldots, L(b_{i-1})$.*

Lemma 17 follows from Lemma 14. Lemma 18 follows from three iterations of Lemma 17. Lemma 19 is proved by induction on n, where the induction step follows from three iterations of Lemma 18 and from Lemma 12. The assertion of Lemma 19 with $n = 2k+1$ implies that $\omega(\mathcal{F}) \geqslant k+1$, which yields Lemma 15. This completes the proof of Theorem 4.

References

[1] E. Ackerman, On the maximum number of edges in topological graphs with no four pairwise crossing edges, *Discrete Comput. Geom.* 41, 365–375, 2009.

[2] P. K. Agarwal, B. Aronov, J. Pach, R. Pollack, M. Sharir, Quasi-planar graphs have a linear number of edges, *Combinatorica* 17, 1–9, 1997.

[3] E. Asplund, B. Grünbaum, On a colouring problem, *Math. Scand.* 8, 181–188, 1960.

[4] P. Brass, W. Moser, J. Pach, *Research Problems in Discrete Geometry*, Springer, 2005.

[5] J. P. Burling, *On coloring problems of families of prototypes*, PhD thesis, University of Colorado, Boulder, 1965.

[6] M. Chudnovsky, A. Scott, P. Seymour, Induced subgraphs of graphs with large chromatic number. V. Chandeliers and strings, arXiv:1609.00314.

[7] J. Fox, J. Pach, Coloring K_k-free intersection graphs of geometric objects in the plane, *European J. Combin.* 33, 853–866, 2012.

[8] J. Fox, J. Pach, Applications of a new separator theorem for string graphs, *Combin. Prob. Comput.* 23, 66–74, 2014.

[9] A. Gyárfás, On the chromatic number of multiple interval graphs and overlap graphs, *Discrete Math.* 55, 161–166, 1985. Corrigendum: *Discrete Math.* 62, 333, 1986.

[10] T. Krawczyk, A. Pawlik, B. Walczak, Coloring triangle-free rectangle overlap graphs with $O(\log \log n)$ colors, *Discrete Comput. Geom.* 53, 199–220, 2015.

[11] T. Krawczyk, B. Walczak, On-line approach to off-line coloring problems on graphs with geometric representations, *Combinatorica*, in press.

[12] M. Lasoń, P. Micek, A. Pawlik, B. Walczak, Coloring intersection graphs of arc-connected sets in the plane, *Discrete Comput. Geom.* 52, 399–415, 2014.

[13] S. McGuinness, On bounding the chromatic number of L-graphs, *Discrete Math.* 154, 179–187, 1996.

[14] S. McGuinness, Colouring arcwise connected sets in the plane I, *Graphs Combin.* 16, 429–439, 2000.

[15] J. Pach, R. Radoičić, G. Tóth, Relaxing planarity for topological graphs, in *More Graphs, Sets and Numbers*, vol. 15 of *Bolyai Soc. Math. Stud.*, 285–300, Springer, 2006.

[16] A. Pawlik, J. Kozik, T. Krawczyk, M. Lasoń, P. Micek, W. T. Trotter, B. Walczak, Triangle-free geometric intersection graphs with large chromatic number, *Discrete Comput. Geom.* 50, 714–726, 2013.

[17] A. Pawlik, J. Kozik, T. Krawczyk, M. Lasoń, P. Micek, W. T. Trotter, B. Walczak, Triangle-free intersection graphs of line segments with large chromatic number, *J. Combin. Theory Ser. B* 105, 6–10, 2014.

[18] A. Rok, B. Walczak, Outerstring graphs are χ-bounded, in *30th Annual Symposium on Computational Geometry (SoCG 2014)*, 136–143, ACM, 2014.

[19] A. Suk, Coloring intersection graphs of x-monotone curves in the plane, *Combinatorica* 34, 487–505, 2014.

[20] A. Suk, B. Walczak, New bounds on the maximum number of edges in k-quasi-planar graphs, *Comput. Geom.* 50, 24–33, 2015.

[21] P. Valtr, Graph drawing with no k pairwise crossing edges, in *5th International Symposium on Graph Drawing (GD 1997)*, vol. 1353 of *Lecture Notes Comput. Sci.*, 205–218, Springer, 1997.

Conflict-Free Coloring of Intersection Graphs

Sándor P. Fekete* Phillip Keldenich*

Abstract

A *conflict-free k-coloring* of a graph $G = (V, E)$ assigns one of k different colors to some of the vertices such that, for every vertex v, there is a color that is assigned to exactly one vertex among v and v's neighbors. Such colorings have applications in wireless networking, robotics, and geometry, and are well studied in graph theory. Here we study the conflict-free coloring of geometric intersection graphs. We demonstrate that geometric objects without fatness properties and size restrictions have intersection graphs with unbounded conflict-free chromatic number. For unit-disk intersection graphs, we prove that it is NP-complete to decide the existence of a conflict-free coloring with one color; we also show that six colors always suffice, using an algorithm that colors unit disk graphs of restricted height with two colors. We conjecture that four colors are sufficient, which we prove for unit squares instead of unit disks.

1 Introduction

Coloring the vertices of a graph is one of the fundamental problems in graph theory, both scientifically and historically. The notion of proper graph coloring can be generalized to hypergraphs in several ways. One natural generalization is *conflict-free coloring*, which asks to color the vertices of a hypergraph such that every hyperedge has at least one uniquely colored vertex. This has applications in wireless communication, where "colors" correspond to different frequencies. The notion can be transported back to simple graphs by considering hypergraphs induced by the neighborhoods of vertices.

In current work with Abel et al. [2], we prove a conflict-free variant of Hadwiger's conjecture, which implies planar graphs have conflict-free chromatic number at most 3; see that paper for a more detailed overview of related work. In the geometric context, motivated by frequency assignment problems, the study of conflict-free coloring of hypergraphs was initiated by Even et al. [5] and Smorodinsky [11]. For disk intersection hypergraphs, Even et al. [5] prove that $\mathcal{O}(\log n)$ colors suffice. For disk intersection hypergraphs with degree at most k, Alon and Smorodinsky [3] show that $\mathcal{O}(\log^3 k)$ colors are sufficient. If every edge of a disk intersection hypergraph must have k distinct unique colors, Horev et al. [8] prove that $\mathcal{O}(k \log n)$ suffice. Moreover, for unit disks, Lev-Tov and Peleg [9] present an $\mathcal{O}(1)$-approximation algorithm for the conflict-free chromatic number. Abam et al. [1] consider the problem of making a conflict-free coloring robust against removal of a certain number of vertices, and prove worst-case bounds for the number of colors required.

Conflict-free coloring also arises in the context of the conflict-free variant of the chromatic art gallery problem, where a simple polygon P has to be guarded by colored guards such that each point in P sees at least one uniquely colored guard. Regarding complexity, Fekete et al. [6] prove that computing the chromatic number is NP-hard in this context. On the positive side, Hoffman et al. [7] give tight bounds for the conflict-free chromatic art gallery problem under rectangular visibility in orthogonal polygons: $\Theta(\log \log n)$ colors are sometimes necessary and always sufficient. For the more common straight-line visibility, Bärtschi et al. [4] prove that $\mathcal{O}(\log n)$ colors always suffice.

2 Preliminaries

In the following, $G = (V, E)$ will denote a graph on $n := |V|$ vertices. For a vertex v, $N(v)$ denotes its *open neighborhood* and $N[v] = N(v) \cup \{v\}$ denotes its *closed neighborhood*. A *conflict-free k-coloring* of a graph $G = (V, E)$ is a coloring $\chi_C : V' \to \{1, \ldots, k\}$ of a subset $V' \subseteq V$ of the vertices of G, such that each vertex v has at least one *conflict-free neighbor* $u \in N[v]$, i.e., a neighbor u whose color $\chi_C(u)$ occurs only once in $N[v]$. The *conflict-free chromatic number* $\chi_C(G)$ is the minimum number of colors required for a conflict-free coloring of G.

A graph G is called *disk graph* iff G is the intersection graph of disks in the plane. A disk graph G is a *unit disk graph* iff G is the intersection graph of disks with fixed radius $r = 1$ in the plane. A graph G is a *unit square graph* iff G is the intersection graph of axis-aligned squares with side length 2 in the plane. A unit disk (square) graph is of *height* h iff G can be modeled by the intersection of unit disks (squares) with center points in $(-\infty, \infty) \times [0, h]$. In the following, when dealing with intersection graphs, we assume that we are given a geometric model. In the case of unit disk and unit square graphs, we identify the vertices of the graph with the center points of the corresponding geometric objects in this model.

*Department of Computer Science, TU Braunschweig, Germany, {s.fekete,p.keldenich}@tu-bs.de

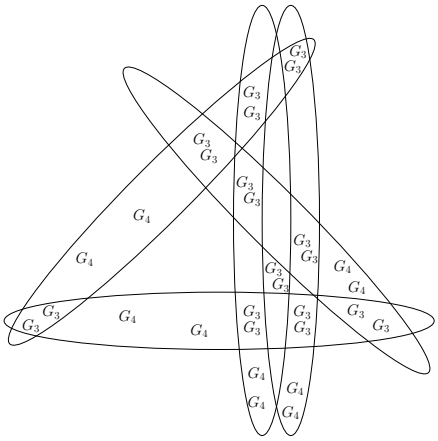


Figure 1: The graph G_5, shown as an intersection graph of ellipses, requires 5 colors.

3 General Objects

For general objects like freely scalable ellipses or rectangles, it is possible to model a complete graph K_n of arbitrary size n such that the following conditions hold: (1) For every object v, there is some non-empty area of v not intersecting any other objects. (2) For every pair of objects v, w, there is a non-empty area common to these objects disjoint from all other objects.

In this case, the conflict-free chromatic number is unbounded, because we can inductively build a family G_n of intersection graphs with $\chi_C(G_n) = n$ as follows. Starting with $G_1 = (\{v\}, \emptyset)$ and $G_2 = C_4$, we construct G_n by starting with a K_n modeled according to conditions (1) and (2). For every object v, we place two scaled-down non-intersecting copies of G_{n-1} into an area covered only by v. For every pair of objects v, w, we place two scaled-down non-intersecting copies of G_{n-2} into an area covered only by v and w. The resulting graph requires n colors, as every vertex of the underlying K_n has to receive a unique color. Figure 1 depicts the construction of G_5 for ellipses.

4 Unit-Disk Graphs

4.1 Complexity: One Color

While it is trivial to decide whether a graph has a regular chromatic number of 1 and straightforward to check a chromatic number of 2, it is already NP-complete to decide whether a conflict-free coloring with a single color exists, even for unit-disk intersection graphs with maximum degree 3. This is a refinement of Theorem 4.1 in Abel et al. [2], which shows the same results for general planar graphs.

Theorem 1 *It is NP-complete to decide whether a unit-disk intersection graph $G = (V, E)$ has a conflict-free coloring with one color.*

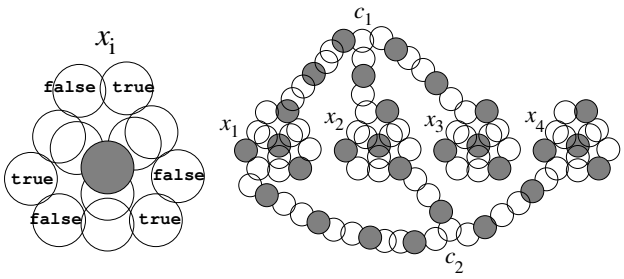


Figure 2: (Left) A variable gadget; note that the central disk must be part of any solution, leaving only the choices labeled `true` and `false` for the other disks. (Right) The overall construction for the instance $(x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_2 \vee \overline{x_4})$.

Proof. We sketch a reduction from Planar 1-in-3-SAT, see Mulzer and Rote [10]. For a given instance I, we build a unit disk intersection graph G_I, in which variables x_i are represented by the gadget shown in Figure 2 (Left), consisting of an exterior cycle of $3n_i$ vertices, for some number $n_i \in \mathbb{N}$, and an auxiliary internal tree. A clause c_j is represented by a single unit disk; we connect it to each of the three involved variable gadget with $3n_\ell$ unit disks, as shown in Figure 2 (Right).

Now a satisfying truth assignment for I induces a conflict-free coloring of G_I with a single color in a straightforward manner. Conversely, in a conflict-free coloring of G with one color, the set $S \subseteq V$ of colored disks is both an independent and a dominating set in G, so any two disks in S must have distance at least 3. This implies that for each exterior cycle in a variable, every third vertex must belong to S, inducing a truth assignment. Similarly, along each connecting path, every third disk must belong to S. As it turns out, no clause disk can be picked, implying that precisely one of its neighbors must be in S; this requires a solution for I. Full details are omitted for lack of space. □

4.2 A Worst-Case Upper Bound: Six Colors

On the positive side, we show that the conflict-free chromatic number of unit disk graphs is bounded by 6. We do not believe this result to be tight. In particular, we conjecture that the number is bounded by 4; in fact, we do not even know an example where two colors are insufficient. One of the major obstacles towards obtaining tighter bounds is the fact that a simple graph-theoretic characterization of unit disk graphs is not available, as recognizing unit disk graphs is complete for the existential theory of the reals. This makes it hard to find unit disk graphs with high conflict-free chromatic number, especially considering the size such a graph would require: The smallest graph with conflict-free chromatic number 3 we know has 30 vertices, and by enumerating all graphs on 12 vertices one can show that at least 13 vertices are necessary, even without the restriction to unit disk graphs.

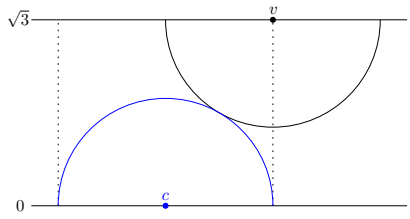


Figure 3: Every colored point c induces a vertical strip of width 2 (dashed lines); all points v within this strip are adjacent to c.

One approach to conflict-free coloring of unit disk graphs is by subdividing the plane into strips, coloring each strip independently. We conjecture the following.

Conjecture 2 *Unit disk graphs of height* 2 *are conflict-free* 2*-colorable.*

If this conjecture holds, every unit disk graph is conflict-free 4-colorable. In this case, one can subdivide the plane into strips of height 2, and then color the subgraphs in all even strips using colors $\{1, 2\}$ and the subgraphs in odd strips using colors $\{3, 4\}$. Instead of Conjecture 2, we prove the following weaker result.

Theorem 3 *Unit disk graphs G of height $\sqrt{3}$ are conflict-free 2-colorable.*

Proof. Given a realization of G consisting of unit disks with center points with y-coordinate in $[0, \sqrt{3}]$, we compute a conflict-free 2-coloring of G using the following greedy approach. We iterate through the disk centers in lexicographical order, choosing a set C of points to be colored. At every iteration, let c be the current and n be the next point. Let C be the set of selected colored points and let $S = N[C]$ be the points that already have a colored neighbor. We select c to be colored iff coloring n instead of c would leave a previous point uncovered, i.e., iff there is a point $c' \notin S$, $c' \leq c$ adjacent to c but not to n. Thus, starting from the leftmost point, we always color the rightmost point that does not leave any previous points without a colored neighbor. We alternatingly assign colors 1 and 2 to the selected points.

In this procedure, any point v is assigned a colored neighbor $w \in N[v]$. This leaves the following three cases. (1) a colored point v is adjacent to another point w of the same color, (2) an uncolored point is adjacent to two or more points of one color and none of the other color, (3) an uncolored point is adjacent to two or more points of both colors.

To this end, we use the following. Each colored point c induces a closed vertical *strip* of width 2 centered around c. As shown in Figure 3, every point v in this strip is adjacent to c. Thus, the horizontal distance between two colored points must be greater than 1. For case (1), assume there was a point v of color 1 adjacent to a point $w > v$ of color 1. This cannot occur, because between v and w, there must be a point x of color 2; therefore, the horizontal distance between v and w must be greater than 2, a contradiction.

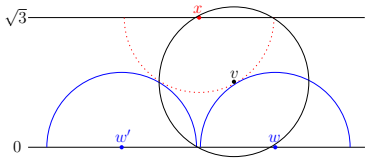


Figure 4: The configuration in case (2); there must be a point x of color 2 adjacent to v.

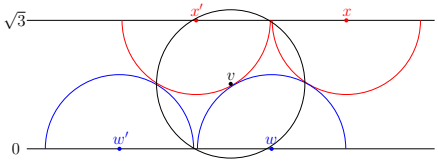


Figure 5: The configuration in case (3); the algorithm would have chosen v or a larger point instead of x'.

Regarding case (2), assume there was an uncolored point v adjacent to two points $w' < v < w$ of color 1. Between points w' and w, there must be a point x of color 2, and v must not be adjacent to x. There are two possible orderings: $w' < v < x < w$ and $w' < x < v < w$. W.l.o.g., let $v < x$; the other case is symmetric. In this situation, the x-coordinates of the points have to satisfy $x(v) < x(x) - 1$, $x(x) < x(w) - 1$, and thus $x(v) < x(w) - 2$ in contradiction to the assumption that v and w are adjacent.

Regarding case (3), assume there was an uncolored point v adjacent to two points $w' < v < w$ of color 1 and two points $x' < v < x$ of color 2. W.l.o.g., assume $w' < x' < v < w < x$ as depicted in Figure 5; the case $x' < w'$ is symmetric. Because w' and v are adjacent, the vertical strip induced by v intersects the strip induced by w'. Thus, there cannot be a point y with $w' < y < v$ not adjacent to w' or v. This is a contradiction to the choice of x': The algorithm would have chosen v or a larger point instead of x'. □

The next Corollary 4 follows by subdividing the plane into strips of height $\sqrt{3}$; Moreover, applying the proof of Theorem 3 to unit square graphs of height 2 instead of $\sqrt{3}$ yields Corollary 5.

Corollary 4 *Unit disk graphs are conflict-free* 6*-colorable.*

Corollary 5 *Unit square graphs of height* 2 *are conflict-free* 2*-colorable. Unit square graphs are conflict-free* 4*-colorable.*

Unfortunately, the proof of Theorem 3 does not appear to have a straightforward generalization to strips of larger height. Further reducing the height to find strips that are colorable with one color is also impossible, because unit interval graphs, which correspond to

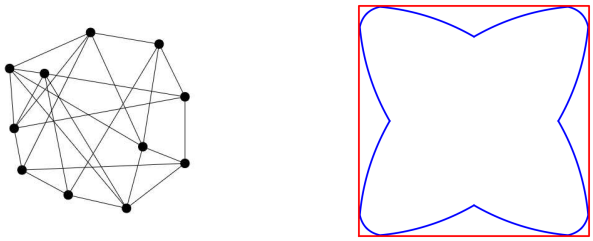


Figure 6: Left: A vertex-minimal graph satisfying (1) and (2). Right: In any unit disk graph G embeddable in a 2×2-square with $\gamma(G) = 3$, no points lie in the depicted area.

unit disk graphs with all centers lying on a line, already may require two colors in a conflict-free coloring; the Bull Graph is such an example. In this case, the bound of 2 is tight: By Theorem 3, unit interval graphs are conflict-free 2-colorable. By adapting the algorithm used in the proof to always choose the interval extending as far as possible to the right without leaving a previous interval uncovered, this can be extended to interval graphs with non-unit intervals.

4.3 Unit-Disk Graphs of Bounded Area

Proving Conjecture 2 is non-trivial, even when all center points lie in a 2×2-square. In this setting, a circle packing argument can be used to establish the sufficiency of three colors. If a unit disk graph with conflict-free chromatic number 3 can be embedded into a 2×2-square, the following are necessary. (1) Every minimum dominating set D has size 3, and every pair of dominating vertices must have a common neighbor not shared with the third dominating vertex. Thus, every minimum dominating set lies on a 6-cycle without chords connecting a vertex with the opposite vertex. (2) G has diameter 2; otherwise, one could assign the same color to two vertices at distance 3.

Using the domination number, one can further restrict the position of the points in the 2×2-square: There is an area in the center of the square, depicted in Figure 6, that cannot contain the center of any disk because this would yield a dominating set of size 2.

The smallest graph satisfying constraints (1) and (2) has 11 vertices and is depicted in Figure 6. It is not a unit disk graph and it is still conflict-free 2-colorable, but every coloring requires at least four colored vertices, proving that coloring a minimum dominating set can be insufficient. This implies that a simple algorithm like the one used in the proof of Theorem 3 will most likely be insufficient for strips of greater height. We are not aware of any unit disk graph satisfying these constraints.

5 Conclusion

There are various directions for future work. In addition to closing the worst-case gap for unit disks (and proving Conjecture 2), it is interesting to study the conflict-free chromatic number of non-unit disk graphs. Other questions include a tight bound for unit square graphs, square intersection graphs of general squares, and a necessary criterion for a family of geometric objects to have intersection graphs with unbounded conflict-free chromatic number.

References

[1] M. A. Abam, M. de Berg, and S.-H. Poon. Fault-tolerant conflict-free colorings. In *Proc. CCCG'08*, pages 13–16, 2008.

[2] Z. Abel, V. Alvarez, E. D. Demaine, S. P. Fekete, A. Gour, A. Hesterberg, P. Keldenich, and C. Scheffer. Three colors suffice: Conflict-free coloring of planar graphs. In *Proc. SODA17*, 2017. To appear.

[3] N. Alon and S. Smorodinsky. Conflict-free colorings of shallow discs. In *Proc. SoCG06*, pages 41–43, 2006.

[4] A. Bärtschi, S. K. Ghosh, M. Mihalák, T. Tschager, and P. Widmayer. Improved bounds for the conflict-free chromatic art gallery problem. In *Proc. SoCG14*, page 144, 2014.

[5] G. Even, Z. Lotker, D. Ron, and S. Smorodinsky. Conflict-free colorings of simple geometric regions with applications to frequency assignment in cellular networks. *SIAM J. Comp.*, 33(1):94–136, 2003.

[6] S. P. Fekete, S. Friedrichs, M. Hemmer, J. B. M. Mitchell, and C. Schmidt. On the chromatic art gallery problem. In *Proc. CCCG14*, pages 1–6, paper 11, 2014.

[7] F. Hoffmann, K. Kriegel, S. Suri, K. Verbeek, and M. Willert. Tight bounds for conflict-free chromatic guarding of orthogonal art galleries. In *Proc. SoCG15*, pages 421–435, 2015.

[8] E. Horev, R. Krakovski, and S. Smorodinsky. Conflict-free coloring made stronger. In *Proc. SWAT10*, volume 6139, pages 105–117, 2010.

[9] N. Lev-Tov and D. Peleg. Conflict-free coloring of unit disks. *Discrete Applied Mathematics*, 157(7):1521–1532, 2009.

[10] W. Mulzer and G. Rote. Minimum-weight triangulation is NP-hard. *J. ACM*, 55(2):11, 2008.

[11] S. Smorodinsky. *Combinatorial Problems in Computational Geometry*. PhD thesis, School of Computer Science, Tel-Aviv University, 2003.

On the Dominating Set Problem in Intersection Graphs*

Mark de Berg† Sándor Kisfaludi-Bak† Gerhard Woeginger‡

Abstract

We give a complete characterization of the parameterized complexity of DOMINATING SET on intersection graphs of 1-dimensonal interval pattern translates, for the case when the pattern is not part of the input. We also describe results about 2-dimensional shapes.

1 Introduction

The DOMINATING SET problem is a model problem in parameterized complexity. This is partially because it has many applications and partially because it has close connections to other basic problems, such as SET COVER and HITTING SET. Moreover, it is one of the few natural problems that are known to be W[2]-complete. The usual definition is the following (we already state the parameterized version). Given a graph G and a parameter k, is there a vertex set S of size k (called dominating set) such that every vertex is either in S or adjacent to a vertex in S?

The natural parameterization by the size of the dominating set can typically yield one of four results in a given class of graphs; DOMINATING SET on the given graph class can be either contained in P (have a polynomial algorithm), or NP-complete; in the latter case it can be expected to be either in FPT, W[1]-complete or W[2]-complete. The problem cannot be on higher levels of the W hierarchy since the DOMINATING SET problem (in short, DS) is W[2]-complete on general graphs [3].

The DS problem has been studied extensively in various intersection graphs. In case of interval graphs, the problem is solvable in polynomial time [2]. However, in case of unit 2-interval graphs—here each vertex corresponds to a pair of disjoint unit intervals on the real line—, the problem is W[1]-complete [4]. In the plane, DS on unit disk graphs has been shown to be W[1]-hard, and in case of unit squares, DS is W[1]-complete [6].

Our contribution. We classify finite 1-dimensional patterns, based on the hardness of the DS problem on intersection graphs of their translates.

We define the problem the following way. Let Q be a fixed subset of $\mathbb{R}$ that consists of a finite number of points and closed intervals, given by their endpoints. We call Q a *pattern*. For a real number x, we define $Q(x) := x + Q$ to be the pattern Q translated by x. We call $Q(x)$ a *Q-translate*. The Q-INTERSECTION DOMINATING SET problem is the following: given a finite set of numbers $x_1, \dots x_n$ and a parameter k, let G be the intersection graph induced by the Q-translates $Q(x_1), \dots, Q(x_n)$. The input is accepted if and only if G has a dominating set of size k. We work in the real RAM model, where we can compute exactly with arbitrary real numbers, and each arithmetic operation is executed in constant time. We define the *distance ratio* of two point pairs $(x_1, x_2), (x_3, x_4) \in \mathbb{R} \times \mathbb{R}$ as $\frac{|x_1 - x_2|}{|x_3 - x_4|}$. Our main theorem is the following.

Theorem 1 *Q-INTERSECTION DOMINATING SET has the following complexity:*

1. *It is in P if there is an interval in Q.*
2. *It is in P if Q is a point pattern and the point distance ratios in Q are rational.*
3. *It is NP-complete and FPT if Q is a point pattern which has at least one irrational distance ratio.*

We also present results on DOMINATING SET on intersection graphs of 2-dimensional shapes. The aim is to get a handle on the boundary between the classes W[1] and W[2]. We show that DS is W[1]-complete for most planar problems where the shapes have a bounded description complexity; otherwise the problem may be W[2]-complete.

2 1-dimensional patterns

In this section, we study the Q-INTERSECTION DOMINATING SET problem in one dimension. If Q contains an unbounded interval, then all translates are intersecting; the intersection graph is a clique and the minimum dominating set is a single vertex. In what follows, we suppose that all intervals in Q are bounded. The *span* between Q is the distance of its leftmost and rightmost point. We prove Theorem 1 by studying each claim separately.

Theorem 2 *Q-INTERSECTION DOMINATING SET can be solved in $O(n^{6w+4})$ time if Q contains an interval, where w is the ratio of the span of Q and the length of the longest interval in Q.*

*This work was supported by NWO grant no. 024.002.003.

†Department of Computer Science, TU Eindhoven. mdberg@win.tue.nl, s.kisfaludi.bak@tue.nl

‡Department of Computer Science, RWTH Aachen University. woeginger@cs.rwth-aachen.de

Note that since Q is a fixed pattern, the value of w does not depend on the input size and so Theorem 2 implies Theorem 1.1.

Next we will prove Theorem 2. We translate Q so that its leftmost endpoint ($\min Q$) is 0. Rescale Q so that its longest interval has length 1 (note that $w = \max Q$ after the rescaling). Consider an intersection graph of the translates of Q. The vertices of $\mathcal{G}$ are $Q(x_i)$ for the given values x_i. We call x_i the *left endpoint* of Q_i. Let $+$ also denote the Minkowski sum of sets: $A + B = \{a + b \mid a \in A, b \in B\}$. If A or B is a singleton, then we omit the braces, i.e., let $a + B$ denote $\{a\} + B$.

Lemma 3 *Let $D \subseteq V(G)$ be a minimum dominating set and let D^* be the set of left endpoints corresponding to the patterns in D. Then for all $y \in \mathbb{R}$ it holds that $|D^* \cap [y, y + w]| \leqslant 3w$.*

Proof. We prove this lemma first for unit interval graphs (where Q consists of a single interval).

▷ Claim. *In any unit interval graph there is a minimum dominating set whose intervals do not overlap.* Proof. Take a minimum dominating set D, and suppose it has two overlapping intervals I_1 and I_2, such that the left endpoint of I_1 lies to the left of the left endpoint of I_2. The set $D \setminus \{I_2\}$ does not dominate every interval. Let H be the set of undominated intervals (the set of intervals that have no neighbor in $D \setminus \{I_2\}$). The intervals in H lie to the right of I_1 (since they were previously intersected by I_2, but they are disjoint from I_1). The rightmost interval $I_H \in H$ intersects all intervals of H, since all intervals of H have their left endpoints in $I_2 \setminus I_1$, an interval of length less than 1. Thus, $(D \setminus \{I_2\}) \cup \{I_H\}$ is a minimum dominating set. Repeating this operation on overlapping intervals terminates because the sum of the left endpoints of the dominating set strictly increases after each step. Therefore, this results in an overlap-free minimum dominating set. ◁

Notice that the lemma immediately follows from this claim in case of unit interval graphs since then $|D^* \cap [y, y + 1]| \leqslant 1 < 3 = 3w$. Let Q be any other pattern, and suppose that $|D^* \cap [y, y + w]| \geqslant 3w + 1$. The patterns starting in $[y, y + w]$ can only dominate patterns with a left endpoint in $[y - w, y + 2w]$, a window of width $3w$. Let H be the set of patterns starting in $[y - w, y + 2w]$. Let U be a unit interval of Q, and consider the intervals corresponding to U in the patterns of H. Notice that U^* is a point set that is also in a window of length $3w$. By the claim above, we know that the interval graph $\mathcal{G}(U)$ defined by U has a dominating set that contains non-overlapping intervals, in particular, a dominating set D_U of size at most $3w$. Since $\mathcal{G}(U)$ corresponds to a spanning subgraph of $\mathcal{G}(H)$, the patterns D_U^H corresponding to D_U in H form a dominating set of $\mathcal{G}(H)$. Thus, $(D \setminus H) \cup D_U^H$ is a dominating set of our original graph that is smaller than D, which contradicts the minimality of D. □

We can now move on to the proof of Theorem 2.

Proof. We give a dynamic programming algorithm. We translate our input so that the left endpoint of the leftmost pattern is 0. Moreover, we can assume that the graph induced by our pattern is connected, since we can apply the algorithm for each connected component separately. The connectivity implies that the left endpoint of the rightmost pattern is at most $(n-1)w$. Let $0 < k \leqslant n$ be an integer and let $G(k)$ be the intersection graph induced by the patterns with left endpoints in $[0, kw]$. Let $\mathcal{I}(k)$ be the set of input patterns with left endpoints in $[(k-1)w, kw)$ and let $S \subseteq \mathcal{I}(k)$. Let $A(k, S)$ be the size of a minimum dominating set D of $G(k)$ for which $D \cap \mathcal{I}(k) = S$. By Lemma 3 it follows that $|S| \leqslant 3w$.

The following recursion holds for $A(k, S)$ if we define $A(0, S) := 0$:

$$A(k,S) = \min \Big\{ A(k-1, S') + |S| \;\Big|\; S' \subset \mathcal{I}(k-1),\ |S'| \leqslant 3w,\ S \cup S' \text{ dominates } \mathcal{I}(k) \Big\}.$$

The total number of subproblems for a fixed value of k is $\sum_{j=0}^{3w} \binom{n}{j} = O(n^{3w})$; thus the number of subproblems is $O(n^{3w+1})$. Computing the value of a subproblem requires looking at $O(n^{3w+1})$ potential subsets S', and $O(n^2)$ time is sufficient to check whether $S \cup S'$ dominates $\mathcal{I}(k)$. Overall, the running time of our algorithm is $O(n^{6w+4})$. □

Theorem 4 *If Q is a point pattern where the distance ratios of any two point pairs of Q are rational, then Q-*INTERSECTION DOMINATING SET *can be solved in polynomial time.*

Proof. We rescale the pattern so that the smallest distance between any pair of points in Q is 1; after this operation all points have rational coordinates. Next, we magnify again by the least common multiple of the divisors in the coordinates, this operation results in a pattern Q' with only integer coordinates.

Let x be the left endpoint of a translate of our pattern given in the input. Consider the connected component $\mathcal{C}$ of $Q'(x)$ in the intersection graph. The union of the patterns in $\mathcal{C}$ are a subset of $x + \mathbb{Z}$. Let P' be the pattern we get from Q' if we replace the point in 0 by the interval $[0, \frac{1}{2}]$. If we replace each pattern in $\mathcal{C}$ by P', the intersection graph remains unchanged, so by Theorem 2, we can compute the minimum DS of this component in polynomial time. By repeating this procedure in each component, we get the desired polynomial algorithm. □

Theorem 5 *If Q is a point pattern where there are two point pairs with an irrational distance ratio, then Q-*INTERSECTION DOMINATING SET *is* NP*-complete.*

Proof. The containment in NP is trivial; we show the hardness by reducing from DS on induced triangular grid graphs. (These are finite induced subgraphs of the triangular grid, which is the graph with vertex set $V = \mathbb{Z}^2$ and edge set $E = \big\{\big((a,b),(a+\alpha,b+\beta)\big) : |\alpha| \leqslant 1, |\beta| \leqslant 1, \alpha \neq \beta\big\}$.) The DS problem is known to be NP-hard on induced grid graphs; triangular grid graphs are not a superclass of grid graphs, so a proof of the NP-hardness of DS in induced triangular grid graphs will be given in the full version.

We show that the infinite triangular grid can be realized as a Q-intersection graph, where the Q-translates are in a bijection with the vertices of the triangular grid. Therefore, any induced triangular grid graph is realized as the intersection graph of the Q-translates corresponding to its vertices.

Rescale Q so that it has span 1. It cannot happen that all the points are rational, because it would make all distance ratios rational as well. Let $x \in Q$ be the smallest irrational point. Consider the intersection of $ax + Q$, $a \in \mathbb{Z}$, with the set $\mathbb{Z} + Q$. We claim that this intersection is non-empty only for a finite number of values $a \in \mathbb{Z}$. Suppose the opposite. Since Q is a finite pattern, there must be a pair $z, z' \in Q$ such that $ax + z = b + z'$ has infinitely many solutions $(a,b) \in \mathbb{Z}^2$. In particular, there are two solutions (a_1,b_1) and (a_2,b_2) such that $a_1 \neq a_2$ and $b_1 \neq b_2$. Subtracting the two equations we get $(a_1 - a_2)x = b_1 - b_2$, which implies $x = \frac{b_1-b_2}{a_1-a_2}$. This is a contradiction since x is irrational.

Let $y = a'x$, where a' is the largest value for which $ax + Q$ intersects $\mathbb{Z} + Q$. It follows that

$$\big\{a \in \mathbb{Z} \,\big|\, (ay + Q) \cap (\mathbb{Z} + Q) \neq \emptyset\big\} = \big\{-1, 0, 1\big\}.$$

Consider the intersection graph induced by the sets $\big\{ay + b + Q \,\big|\, (a,b) \in \mathbb{Z}^2\big\}$. The above shows that a fixed translate $ay + b + Q$ is not intersected by the translates $(a+\alpha)y + (b+\beta) + Q$ if $|\alpha| \geqslant 2$. It is easy to see that $|\beta| \geqslant 2$ does not lead to an intersection either. Also note that $\alpha = \beta = \pm 1$ does not give an intersection; however all the remaining cases are intersecting, i.e., if $(\alpha,\beta) \in \big\{(-1,0),(-1,1),(0,-1),(0,0),(0,1),(1,-1),(1,0)\big\}$ then $(a+\alpha)y + (b+\beta) + Q$ intersects $ay + b + Q$. Thus, the intersection graph induced by $\big\{ay + b + Q \,\big|\, (a,b) \in \mathbb{Z}^2\big\}$ is a triangular grid. □

Theorem 6 *If Q is a point pattern that has point pairs with an irrational distance ratio, then Q-*INTERSECTION DOMINATING SET *has an* FPT *algorithm (parameterized by solution size).*

Proof. The intersection graph of a point pattern containing t points has maximum degree $\binom{t}{2}$, thus we are looking for a dominating set in a graph of bounded degree. Hence, a straightforward branching approach gives an FPT algorithm. Choose any undominated vertex v; either v or one of its at most $\binom{t}{2}$ neighbors is in the dominating set, so we can branch $1 + \binom{t}{2}$ ways. If there are no undominated vertices after choosing k vertices, then we have found a solution. This branching algorithm has depth k, with linear time required at each branching, so the total running time is $O\big(t^{2k}(|V| + |E|)\big)$. □

Remark. In our handling of the problem, the pattern was part of the problem definition. Making the pattern part of the input leads to an NP-complete problem: Theorem 5 can be adapted to this scenario, even when changing to rational inputs and patterns and word RAM. This would therefore hide the tractability claims of Theorems 2 and 4; this is our justification for using the real RAM model.

If the pattern is part of the input, then there are open questions left to answer.

Open question. Let Q be the pattern defined by two unit intervals on a line at distance ℓ. Is there an FPT algorithm (with parameter $k+\ell$) on intersection graphs defined by translates of Q, that can decide if such a graph has a DOMINATING SET of size k? (It can be shown that this problem is NP-complete, and Theorem 8 below shows that it is in W[1].)

3 Two dimensional shapes: W[1] vs. W[2]

In this section we show that DOMINATING SET on 2-dimensional intersection graphs is contained in W[1] if the shapes have a constant size description. Such a containment can be proven by providing a nondeterministic algorithm that contains two phases in this order:

1. an FPT time deterministic preprocessing
2. a nondeterministic phase where the number of steps is dependent only on the parameter.

The exact formulation can be found in [5]. Note that with the exception of the following theorem, all other proofs are omitted in this section, but will be provided in the full version of the paper.

Theorem 7 *The* DS *problem on unit disk graphs is contained in* W[1].

Proof. For a subset $D \subseteq P$, let $\mathcal{C}_2(D)$ and $\mathcal{D}_2(D)$ be the set of circles and disks of radius 2 centered at the points of D respectively. (Note that D is a dominating set if and only if $\bigcup \mathcal{D}_2(D)$, the union of the the disks in $\mathcal{D}_2(D)$ covers all points in P.) Shoot a vertical ray up and down from each of the $O(k^2)$ intersection points between the circles of $\mathcal{C}_2(D)$, and also from the

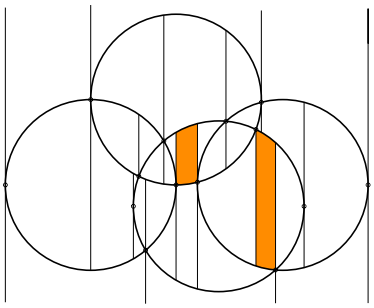


Figure 1: Two faces of a vertical decomposition.

leftmost and rightmost point of each circle. Each ray is continued until it hits a circle (or to infinity). The arrangement we get is a *vertical decomposition* [1] (see Fig. 1). Each face of this decomposition is defined by at most four circles (including degenerate and lower dimensional faces).

In our preprocessing phase, we compute all potential faces of a vertical decomposition of any subset $D \subseteq P$ by looking at all 4-tuples of circles from $\mathcal{C}_2(P)$. We create a lookup table that contains the number of input points covered by each potential face in $O(n^4)$ time.

Next, using nondeterminism we guess k integers, representing the points of our solution; let D be this point set. The rest of the algorithm deterministically checks if D is dominating. We need to compute the vertical decomposition of $\mathcal{C}_2(D)$; this can be done in $O(k^2)$ time [1]. Finally, for each of the $O(k^2)$ resulting faces of $\bigcup \mathcal{D}_2(D)$, we can get the number of input points covered from the lookup table in constant time. We accept if these numbers sum to n. □

In order to state the general version of this theorem, we introduce semialgebraic sets. A *semialgebraic set* is a subset of $\mathbb{R}^d$ obtained from a finite number of sets of the form $\{x \in \mathbb{R}^d \mid g(x) \geqslant 0\}$, where g is a d-variate polynomial with integer coefficients, by Boolean operations (unions, intersections, and complementations). Let $\Gamma_{d,\Delta,s}$ denote the family of all semialgebraic sets in $\mathbb{R}^d$ defined by at most s polynomial inequalities of degree at most Δ each. If d, Δ, s are all constants, we refer to the sets in $\Gamma_{d,\Delta,s}$ as constant-complexity semialgebraic sets.

A family $\mathcal{F}$ of shapes in Euclidean space is called *nice* if there are a constant d and mappings α and $\mathfrak{A}$ defined on $\mathcal{F}$ with the following properties:

- $\alpha(s)$ is a point in $\mathbb{R}^d$ and $\mathfrak{A}(s)$ is a constant-complexity semialgebraic set in $\mathbb{R}^d$, for any $s \in \mathcal{F}$
- s intersects s' if and only if $\alpha(s) \in \mathfrak{A}(s')$ for any two $s, s' \in \mathcal{F}$.

Example Balls in $\mathbb{R}^3$. Let $\alpha : \mathcal{F} \to \mathbb{R}^4$ be the function which assigns the point (x_1, x_2, x_3, r) to the ball s with center (x_1, x_2, x_3) and radius r. The ball (x_1, x_2, x_3, r) is intersected by the ball s' with parameters (x'_1, x'_2, x'_3, r') if and only if $(x_1 - x'_1)^2 + (x_2 - x'_2)^2 + (x_3 - x'_3)^2 - (r + r')^2 \leqslant 0$. Let $\mathfrak{A}(s')$ be the set of points $(x_1, x_2, x_3, r) \in \mathbb{R}^4$ that satisfy the above polynomial inequality.

A class $\mathcal{S}$ of finite graphs is called *a nice intersection graph class* if there is a nice shape family $\mathcal{F}$ such that all $\mathcal{G} \in \mathcal{S}$ is an intersection graph of some of the shapes in the family, i.e., for all $\mathcal{G} \in \mathcal{S}$ there is a function $\varphi : V(\mathcal{G}) \to \mathcal{F}$ such that $(uv) \in E(\mathcal{G}) \iff \varphi(u) \cap \varphi(v) \neq \emptyset$.

Theorem 8 *The* DS *problem is in* W[1] *for the intersection graph defined by a given collection of n shapes from a nice shape family $\mathcal{F}$.*

We have the following hardness result, stated in a slightly specialized form to avoid long definitions. The last theorem shows that if we allow each shape to have a longer description (polynomial in the number of shapes), then the problem seems to become harder.

Theorem 9 *The* DS *problem is* W[1]*-hard for intersection graphs of translates of a simple polygon.*

Theorem 10 *The* DS *problem is* W[2]*-complete for intersection graphs of convex polygons (where each polygon is defined separately in the input).*

References

[1] M. d. Berg, O. Cheong, M. v. Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 3rd edition, 2008.

[2] M.-S. Chang. Efficient algorithms for the domination problems on interval and circular-arc graphs. *SIAM J. Comput.*, 27(6):1671–1694, 1998.

[3] R. G. Downey and M. R. Fellows. Fixed-parameter tractability and completeness I: Basic results. *SIAM J. Comput.*, 24(4):873–921, 1995.

[4] M. R. Fellows, D. Hermelin, F. A. Rosamond, and S. Vialette. On the parameterized complexity of multiple-interval graph problems. *Theor. Comput. Sci.*, 410(1):53–61, 2009.

[5] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2006.

[6] D. Marx. Parameterized complexity of independence and domination on geometric graphs. In *Proceedings of IWPEC 2006, Zürich*.

Finding Triangles and Computing the Girth in Disk Graphs*

Haim Kaplan† Wolfgang Mulzer‡ Liam Roditty§ Paul Seiferth‡

Abstract

Let $S \subset \mathbb{R}^2$ be a set of n point *sites*, where each $s \in S$ has an *associated radius* $r_s > 0$. The *disk graph* $\mathrm{D}(S)$ of S is the graph with vertex set S and an edge between two sites s and t if and only if $|st| \leq r_s + r_t$, i.e., if the disks with centers s and t and radii r_s and r_t, respectively, intersect. Disk graphs are useful to model sensor networks.

We study the problems of finding triangles and of computing the girth in disk graphs. These problems are notoriously hard for general graphs, but better solutions exist for special graph graph classes, such as planar graphs. We obtain similar results for disk graphs. In particular, we observe that the unweighted girth of a disk graph can be computed in $O(n \log n)$ worst-case time and that a shortest (Euclidean) triangle in a disk graph can be found in $O(n \log n)$ expected time.

1 Introduction

Disk graphs are geometrically defined graphs that show up in many applications and are defined as follows. We are given a set $S \subset \mathbb{R}^2$ of n point sites in the plane, such that each $s \in S$ has an *associated radius* $r_s > 0$. Let the *disk corresponding to* s, denoted by D_s, be the closed disk with center s and radius r_s. The *disk graph for* S, $\mathrm{D}(S)$, is the graph with vertex set S in which two sites s and t are connected by an (undirected) edge if any only if $D_s \cap D_t \neq \emptyset$ (or equivalently if any only if $|st| \leq r_s + r_t$). In a *weighted disk graph*, the weight of an edge (s, t) is equal to $|st|$: the Euclidean distance between s and t.

Even though disk graphs may be dense, it turns out that many algorithmic problems can be solved faster in disk graphs than in general graphs. For example, we can compute the BFS-tree from any given site in an unweighted disk graph in $O(n \operatorname{polylog}(n))$ expected time [2,9], and we can approximate the shortest paths distances in a weighted disk graph by a sparse *spanner* that can be constructed in $O(n \operatorname{polylog}(n))$ expected time [6, 9].

We give fast and simple algorithms for two classic problems when restricting the input to disk graphs. These problems are known to be challenging in general graphs. Our first problem is to determine whether a given graph contains a triangle (i.e., a complete subgraph on three vertices), and, if so, to find a triangle that minimizes the sum of its edge lengths (in the weighted case). For general unweighted graphs, the problem can be solved in $O(n^\omega)$ time using fast matrix multiplication [7, 8] (where $\omega < 2.37287$ is the matrix multiplication exponent). However, if we insist on *combinatorial* algorithms that do not use algebraic techniques, the fastest known algorithm runs in $O(n^3 \operatorname{polyloglog}(n)/\log^4 n)$ time [11]. In planar graphs, the problem can be solved in $O(n)$ time [4]. In the weighted case for general graphs, progress has been made only very recently [10].

The second problem is computing the *girth*, which is the length of a shortest cycle in G. Again, the best result for general unweighted graphs relies on fast matrix multiplication [8], while for planar graphs, the unweighted girth can be computed in linear time [4].

As we will see, these problems are easier for disk graphs. Indeed, finding triangles in disk graphs is almost a trivial problem. By using the intimate connection between disk graphs and planar graphs and some known results for planar graphs, we can extend our observations regarding triangles in disk graphs to an algorithm that computes the unweighted girth of disk graphs. Computing the length of a shortest triangle is a bit more difficult, but we can exploit the geometric structure of disk graphs to solve the decision version of this problem in $O(n \log n)$ time. Then we use Chan's framework for randomized geometric optimization algorithms [3] to solve the optimization problem in the same expected time.

2 Computing the Unweighted Girth

First, we consider the unweighted girth in disk graphs. Given a disk graph $\mathrm{D}(S)$, we would like to find a cycle in $\mathrm{D}(S)$ with the smallest number of edges. It turns out that this problem is closely related to the problem of computing the girth in planar graphs. The following simple property of disk graphs is the key to our algorithm. It has been observed before by Evens et al. [5]. For completeness, we include a proof.

Lemma 1 *Let $\mathrm{D}(S)$ be a disk graph that is not plane. (By this we mean that the embedding obtained by connecting each pair of adjacent sites s and t by a*

*Supported by GIF project 1161&DFG project MU/3501-1.

†Tel Aviv University, Israel. haimk@post.tau.ac.il

‡Institut für Informatik, Freie Universität Berlin, Germany {mulzer,pseiferth}@inf.fu-berlin.de

§Bar Ilan University, Israel. liamr@macs.biu.ac.il

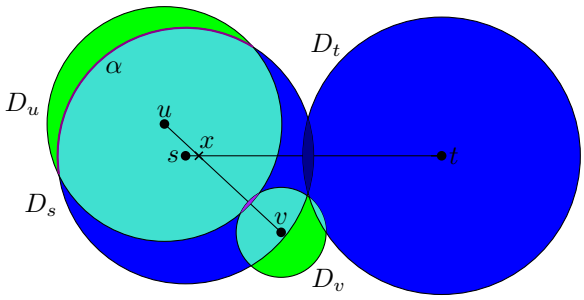


Figure 1: If $\mathrm{D}(S)$ is not plane, then three disks intersect in a common point.

straight segment between s and t, has at least one pair of segments that cross in their relative interior.) Then, there are three sites whose disks intersect in a common point.

Proof. Suppose that the two edges st and uv intersect at a point x. The sites s, t, u, and v are pairwise distinct, and without loss of generality, we may assume that: (i) $x \in D_s \cap D_u$; (ii) $r_s \geq r_u$; (iii) the edge st lies on the x-axis, with s to the left of t; and (iv) the site u lies above the x-axis, and the site v lies below the x-axis; see Figure 1.

Now consider the arc $\alpha = D_u \cap \partial D_s$. If $\alpha = \emptyset$, we are done, since then $D_u \subset D_s$, and $D_s \cap D_u \cap D_v \neq \emptyset$. Furthermore, α must contain a point below the x-axis, because v lies below the x-axis and otherwise we would hence have $D_s \cap D_u \cap D_v \neq \emptyset$. Since u is above the x-axis and since $r_s \geq r_u$, α must intersect the x-axis. This intersection must be to the left of s, since otherwise we would have $D_s \cap D_t \cap D_u \neq \emptyset$. Again since $r_s \geq r_u$, it follows that u lies to the left of s. Now we see that α contains no point that lies below the x-axis and to the right of s. From this and the fact that x lies to the right of s, we can conclude that $uv \cap \partial D_u \in D_s$, and hence $D_s \cap D_u \cap D_v \neq \emptyset$. □

By Lemma 1, we know that if $\mathrm{D}(S)$ is not plane, then the girth is 3. On the other hand, if $\mathrm{D}(S)$ is plane, and if the embedding is available, the girth can be found in $O(n)$ time. More precisely, we can use the following result by Chang and Lu [4].

Theorem 2 (Theorem 1.1 in [4]) *Let G be an unweighted planar graph with n vertices. The girth of G can be computed in $O(n)$ time.*

Combining Lemma 1 and Theorem 2, we immediately obtain a fast algorithm for computing the girth in disk graphs.

Theorem 3 *Let $\mathrm{D}(S)$ be a disk graph with n vertices. We can compute the unweighted girth of $\mathrm{D}(S)$ in $O(n \log n)$ worst-case time.*

Proof. We use a standard sweep-line algorithm to compute the arrangement of the disks corresponding to S [1]. The intersections of the disk boundaries are reported one by one, and the total time to report the first m intersections is $O(n \log n + m \log n)$ [1]. Since every edge in $\mathrm{D}(S)$ corresponds to two unique intersection points, it follows that as soon as $6n - 13$ intersection points have been reported, it must be the case that $\mathrm{D}(S)$ is not plane, and hence, by Lemma 1, the girth is 3. Otherwise, we obtain an explicit representation of $\mathrm{D}(S)$, and we can test in $O(n)$ time whether is it plane. If this is not the case, we again output that the girth is 3. Finally, if $\mathrm{D}(S)$ is plane, we determine the unweighted girth in $O(n)$ time using Theorem 2. □

3 Finding a Shortest Triangle

Now we consider the situation where each edge in $\mathrm{D}(S)$ is weighted according to its Euclidean length. We would like to find a shortest triangle in $\mathrm{D}(S)$, i.e., a triangle that minimizes the sum of its edge lengths. First, we focus on the decision problem: given a parameter $W > 0$, does $\mathrm{D}(S)$ contain a triangle with weight at most W? Once an algorithm for the decision problem is available, a solution for the optimization problem will follow through a straightforward application of Chan's randomized framework for geometric optimization problems [3]. In the end, we will prove the following theorem.

Theorem 4 *Let $\mathrm{D}(S)$ be a disk graph with n vertices, where the edges are weighted according to their Euclidean lengths. We can compute a shortest triangle in $\mathrm{D}(S)$ in $O(n \log n)$ expected time, if it exists.*

3.1 The Decision Problem

Let $S \subset \mathbb{R}^2$ and a weight $W > 0$ be given. To decide if $\mathrm{D}(S)$ contains a triangle with weight at most W, we proceed as follows. We classify the sites as *small* and *large*, depending on their associated radius. This yields four possibilities for our desired triangle. To investigate each such possibility, we use a grid whose cells have a diameter proportional to W. First, we consider only triangles where all three vertices are small and lie in the same grid cell. This can be done using the tools from the previous section. If no cell contains such a triangle, we can show that the graph must be sparse and that we need to check only few further triangles. Details follow.

The Four Cases. Set $\ell = W/(12\sqrt{2})$. We say that a site $s \in S$ is *small*, if $r_s < \ell$, and *large*, otherwise. Depending on the number of small and large vertices, we classify the triangles in $\mathrm{D}(S)$ into four types:

(SSS) 3 small vertices, 0 large vertices

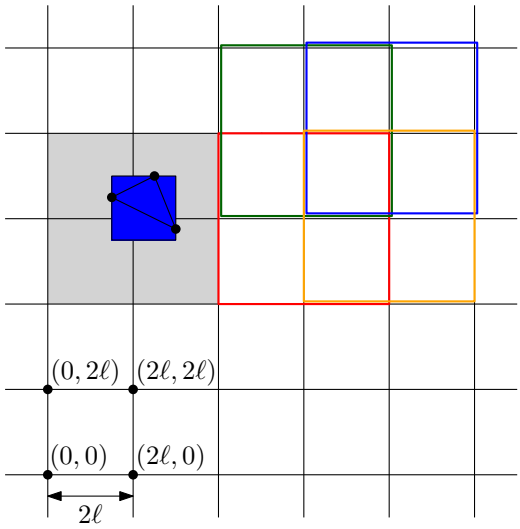


Figure 2: The four shifted grids, with a cell from each grid shown in red, orange, green, and blue, respectively. Every square with side length at most 2ℓ is wholly contained in a single grid cell.

(SSL) 2 small vertices, 1 large vertex

(SLL) 1 small vertex, 2 large vertices,

(LLL) 0 small vertices, 3 large vertices.

Next, we define an appropriate grid that helps us to detect triangles of type (SSS).

The Grid. Let G_1 be the grid whose cells are pairwise disjoint, axis-parallel squares with diameter $W/3$. The cells of G_1 partition the plane, and G_1 is aligned such that the origin $(0, 0)$ is a vertex of G_1. Observe that the cells of G_1 have side length $4\ell = W/(3\sqrt{2})$. We want to ensure that any triangle of type (SSS) in $\mathrm{D}(S)$ is completely contained in a single grid cell. For this, we make three copies G_2, G_3, and G_4 of G_1 and we shift them by 2ℓ (half the side length of a cell) in x-direction, in y-direction, and in both x- and y-direction, respectively. In other words, G_2 has $(2\ell, 0)$ as a vertex, G_3 has $(0, 2\ell)$ as a vertex, and G_4 has $(2\ell, 2\ell)$ as a vertex, see Figure 2.

Lemma 5 *Let Δ be a triangle formed by three vertices $a, b, c \in \mathbb{R}^2$ such that each edge of Δ has length at most 2ℓ. Then, there is a cell $\sigma \in G_1 \cup G_2 \cup G_3 \cup G_4$ with $a, b, c \in \sigma$.*

Proof. By assumption, we can enclose Δ with a square of side length 2ℓ. By construction, this square must be completely contained in cell of one of the four grids, see Figure 2. □

It follows immediately that any triangle of type (SSS) lies in a single grid cell, as desired.

Finding Triangles Inside Grid Cells. Next, we search for triangles that are completely contained in one grid cell. (These are not necessarily triangles of type (SSS).) For this, we go through all nonempty grid cells $\sigma \in \bigcup_{i=1}^{4} G_i$, and we search for a triangle in the disk graph $\mathrm{D}(S \cap \sigma)$ induced by the sites lying in σ. This can be done using Theorem 3. Since the grid cells have diameter $W/3$, any such triangle has weight at most W. Thus, we can return YES if a triangle is found. If we do not find any triangles, we can conclude by Lemma 5 that $\mathrm{D}(S)$ has no triangle of type (SSS). Since each site lies in a constant number of grid cells, and since we can compute the grid cells for a given site in $O(1)$ time, the total running time for this step is $O(n \log n)$. A simple volume argument yields the following lemma.

Lemma 6 *Let $\sigma \in \bigcup_{i=1}^{4} G_i$ be a nonempty grid cell, and suppose that σ does not contain a triangle. Then, σ contains $O(1)$ large sites.*

Proof. Suppose that σ contains at least 19 large sites. We partition σ into 3×3 congruent squares with side length $(4/3)\ell$. Each square has diameter $(4\sqrt{2}/3)\ell < 2\ell$, and by the pigeonhole principle, there is at least one square τ with at least $\lceil 19/9 \rceil = 3$ large sites. Since the associated radius of a large site is at least ℓ, the large sites in τ form a triangle in σ, contrary to our assumption. □

Triangles of Type (LLL). Now suppose that no triangle from $\mathrm{D}(S)$ is contained in a single grid cell. To find triangles of type (LLL), we iterate through all large sites $s \in S$. Let $\sigma \in G_1$ be the grid cell containing s. We define the *neighborhood* $N(\sigma)$ of σ as the 5×5 block of cells in G_1 that is centered at σ. Since the diameter of a grid cell is $W/3$, any pair $u, v \in S$ of sites that form a triangle with s of weight at most W must be contained in $N(\sigma)$. Let $S_\ell \subseteq S$ denote the large sites. By Lemma 6, we have $|N(\sigma) \cap S_\ell| = O(1)$. Thus, we can check in constant time whether s participates in a triangle of type (LLL). Hence, the total time to detect triangles of type (LLL) is $O(n)$.

Triangles of Type (SLL). This case is similar to the algorithm for triangles of type (LLL). This time, we iterate over all small sites $s \in S$. For each small $s \in S$, we check all pairs of large vertices contained in $N(\sigma) \cap S_\ell$, where $\sigma \in G_1$ is the cell containing s. This requires $O(n)$ time.

Triangles of Type (SSL). Now, consider an edge $e = ab \in D(S)$ between two small sites a and b. The edge e has length at most 2ℓ, and by construction it is completely contained in a single grid cell $\sigma \in \bigcup_{i=1}^{4} G_i$. To check if e participates in a triangle of

type (SSL) with weight at most W, we check all $O(1)$ large vertices in $S_\ell \cap N(\sigma)$.

We repeat this process for all edges of $\mathrm{D}(S)$ that lie in a single grid cell, and we claim that this requires $O(n \log n)$ time. Indeed, we know that no grid cell $\sigma \in \bigcup_{i=1}^{4} G_i$ contains a triangle (otherwise, we would have detected it previously). Then, by Lemma 1, it follows that $\mathrm{D}(S \cap \sigma)$ is plane, for all grid cells $\sigma \in \bigcup_{i=1}^{4} G_i$. In particular, $\mathrm{D}(S \cap \sigma)$ has $O(|S \cap \sigma|)$ edges and can be computed in time $O(|S \cap \sigma| \log |S \cap \sigma|)$. Since each vertex is contained in $O(1)$ grid cells, the total time to detect triangles of type (SSL) is $O(n \log n)$, as claimed.

Wrapping up. We summarize the previous discussion with the next lemma.

Lemma 7 *Let $\mathrm{D}(S)$ be a disk graph with n vertices, and let $W > 0$. We can decide in $O(n \log n)$ worst-case time whether $\mathrm{D}(S)$ contains a triangle of weight at most W.*

Finally, to solve the optimization problem, we employ the following general lemma due to Chan [3]. Let Π be a *problem space*, and for a problem $P \in \Pi$, let $w(P) \in \mathbb{R}$ be its *optimum* and $|P| \in \mathbb{N}$ be its *size*.

Lemma 8 (Lemma 2.1 in [3]) *Let $\alpha < 1$, $\varepsilon > 0$, and $r \in \mathbb{N}$ be constants, and let $\delta(\cdot)$ be a function such that $\delta(n)/n^{\varepsilon}$ is monotone increasing in n. Given any optimization problem $P \in \Pi$ with optimum $w(P)$, suppose that within time $\delta(|P|)$, (i) we can decide whether $w(P) < t$, for any given $t \in \mathbb{R}$, and (ii) we can construct r subproblems $P_1, \ldots, P_r$, each of size at most $\lceil \alpha |P| \rceil$, so that*

$$w(P) = \min\{w(P_1), \ldots, w(P_r)\}.$$

Then, we can compute $w(P)$ in total expected time $O(\delta(|P|))$.

For the first condition of Lemma 8, we use Lemma 7. For the second condition we construct four subsets $S_0, \ldots, S_3$ of S as follows: we enumerate the sites in S as $S = \{s_1, \ldots, s_n\}$, and we put each site s_i into all sets S_j with $j \not\equiv i \pmod 4$. Then, for any three sites $a, b, c \in S$, there is at least one subset S_j with $a, b, c \in S_j$. Hence, applying Lemma 8 with $\alpha = 3/4$, $\varepsilon = 1$, $r = 4$, and $\delta = O(n \log n)$ establishes Theorem 4.

4 Conclusion

Once again, disk graphs prove to be a simple and useful graph model where difficult algorithmic problems admit faster solutions. It would be interesting to find a *deterministic* $O(n \log n)$ time algorithm for finding a shortest triangle in a disk graph. Also, we are currently working on extending our results to the girth problem in *weighted* disk graphs and in directed transmission graphs.

Acknowledgments. We like to thank Günther Rote for helpful comments.

References

[1] M. de Berg, O. Cheong, M. van Kreveld, and M. H. Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, third edition, 2008.

[2] S. Cabello and M. Jejčič. Shortest paths in intersection graphs of unit disks. *Comput. Geom. Theory Appl.*, 48(4):360–367, 2015.

[3] T. M. Chan. Geometric applications of a randomized optimization technique. *Discrete Comput. Geom.*, 22(4):547–567, 1999.

[4] H.-C. Chang and H.-I. Lu. Computing the girth of a planar graph in linear time. *SIAM J. Comput.*, 42(3):1077–1094, 2013.

[5] W. S. Evans, M. van Garderen, M. Löffler, and V. Polishchuk. Recognizing a DOG is hard, but not when it is thin and unit. In *Proc. 8th FUN*, pages 16:1–16:12, 2016.

[6] M. Fürer and S. P. Kasiviswanathan. Spanners for geometric intersection graphs with applications. *J. of Computational Geometry*, 3(1):31–64, 2012.

[7] F. Le Gall. Powers of tensors and fast matrix multiplication. In *Proc. 39th Internat. Symp. Symbolic and Algebraic Comput. (ISSAC)*, pages 296–303, 2014.

[8] A. Itai and M. Rodeh. Finding a minimum circuit in a graph. *SIAM J. Comput.*, 7(4):413–423, 1978.

[9] H. Kaplan, W. Mulzer, L. Roditty, P. Seiferth, and M. Sharir. Dynamic planar Voronoi diagrams for general distance functions and their algorithmic applications. In *Proc. 28th SODA*, pages 2495–2504, 2017.

[10] L. Roditty and V. Vassilevska Williams. Minimum weight cycles and triangles: Equivalences and algorithms. In *Proc. 52nd Annu. IEEE Sympos. Found. Comput. Sci. (FOCS)*, pages 180–189, 2011.

[11] H. Yu. An improved combinatorial algorithm for Boolean matrix multiplication. In *Proc. 42nd Internat. Colloq. Automata Lang. Program. (ICALP)*, pages 1094–1105, 2015.

Geomasking through Perturbation, or Counting Points in Circles

Maarten Löffler* Jun Luo† Rodrigo I. Silveira‡

Abstract

Motivated by a technique in privacy protection, in which n points are randomly perturbed by at most a distance r, we study the following problem: Given n points and m circles in the plane, what is the maximum r such that the number of points included in each circle does not change? We also consider a more general question, where we allow the number of points included in each circle to change by a certain factor. We discuss several algorithms for the problems, analyze what parameters of the input influence their running times, and consider a special case where the circles are aligned on a grid, which has important applications.

1 Introduction

The increasing use of mobile communication devices embedded with positioning capabilities (e.g., GPS) is generating vast amounts of geographical data. How to explore such data to find useful information is a central question in spatial data mining. Through spatial data mining, we can discover interesting spatial patterns such as spatial outliers, spatial co-location rules and spatial clusters [10]. However, this also raises the issue of confidentiality, since the results of spatial data mining could reveal personal information such as home or work addresses. Geomasking, or geographic masking, is a set of techniques aimed at modifying location data to make it difficult to recover the original coordinates. For example, the area in which geomasking is most widespread is health, since public health researchers that use data from individuals must employ some kind of geomasking to comply with ethical and legal requirements [14].

The goals of data mining and privacy protection are often conflicting with each other and cannot be satisfied simultaneously. On the one hand, we want to mine meaningful results from the dataset which requires the dataset to be accurate. On the other hand, in order to protect privacy, we usually need to modify the original data. There are two widely adopted forms of geomasking:

- aggregation (also known as *cloaking*), which hides a user's location in a larger region.
- random perturbation (also known as *dithering* or *jittering*), accomplished by changing the original location to a new nearby random location.

In this paper we are interested in random perturbation methods [9, 13], which have a long history in statistical disclosure control due to their simplicity, efficiency and ability to preserve statistical information. Random perturbation consists in displacing each point by a randomly determined distance in a randomly determined direction, with respect to its original location. The maximum displacement is called *perturbation threshold*. Most previous work on perturbation uses the same transformation function for all points, although sometimes it can vary according to local parameters, such as population density [14]. There have been several studies that confirm that the accuracy of spatial analysis results depends heavily on the threshold value [6]: the larger the threshold, the more that spatial patterns get disrupted. Despite widespread agreement on the need to understand better the effects of geomasking on the spatial properties of a set of locations, few studies have addressed this (see [14] for a recent survey). All studies to date deal with one or more concrete data sets, and compare how the spatial patterns of a set of locations change after geomasking. Typical ways to measure spatial patterns include techniques such as the cross-K function, SatScan, and very often, kernel density estimation (KDE).

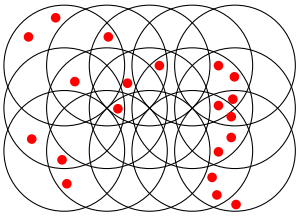
The geometric problems studied in this paper are motivated from the use of KDE [12], a standard tool used in many point-based clustering methods. In its simplest formulation, KDE works as follows. First the plane is divided by a grid consisting of equal-size squares (called *cells*). Then equal-size circles (whose radius is known as *bandwidth*) around each cell center are drawn. Finally, the number of points inside each circle is counted and assigned to the corresponding cell as its density (see Figure 1). Interestingly, previous studies trying to understand the effect of the perturbation threshold on spatial patterns obtained from KDE (e.g., [6, 11]), treat KDE as a black box, and essentially try to find empirically a combination of values for the threshold and bandwidth that give good results.

In this work, we approach the problem in the opposite direction. Knowing that the main criterion

*Dept. of Information and Computing Science, Universiteit Utrecht, m.loffler@uu.nl

†Lenovo Big Data Lab, Hong Kong, jluo1@lenovo.com

‡Dept. de Matemàtiques, Universitat Politècnica de Catalunya, rodrigo.silveira@upc.edu



4	5	4	5	5
4	3	3	8	7
3	3	1	6	8

Figure 1: A set of points in the plane covered by a set of unit-circles arranged on a grid. This is the typical setting in which KDE works, producing a density table like the one shown on the right.

in KDE is point-in-circle containment, we want to find the maximum threshold that preserves the circle-memberships of the points, or one that guarantees only a few changes. We study random permutations where each original point p is replaced by a randomly computed point p' that lies inside a disk of radius r (i.e., the perturbation threshold) and centered at p. Our goal is to find the largest r that preserves all point-circle incidences, or changes them only by a small amount. More precisely, we study the following two geometric problems:

Problem 1 *Given n points and m circles in the plane, compute the maximum perturbation threshold such that the number of points in each circle does not change.*

Problem 2 *Given n points and m circles in the plane, and a parameter δ, compute the maximum perturbation threshold such that the number of points included in each circle changes by at most a factor δ.*

Contributions We study these two problems with the application of geomasking in mind. Our first contribution is to draw attention to two natural and simple geometric problems that, as far as we know, have not been studied in computational geometry before. We identify several input parameters, properties of the given points and circles, that can have an influence in practice in the running time of the algorithms. Then we discuss different algorithms for the problems that use several well-known geometric techniques. We finish by stating an intriguing open problem that can have important practical implications, related to counting points inside unit-disk circles centered on a grid.

Previous work To the best of our knowledge, the only previous work that follows our approach is [7], where the points are allowed to move randomly such that the Delaunay triangulation (DT) of the perturbed points is the same as that of the original points. The motivation to preserve the DT is to maintain topological relationships between the points.

2 Problem description and parameter analysis

Several algorithms for the problems will be discussed in the next section. However, it is easy to observe that the inherent complexity of the problem comes from the possible distances between points and circles, which are $\Theta(nm)$ in total. Given the practical importance of the problems, we are interested in understanding what parameters of the input influence these running times, and in showing that in specific cases we can do better.

We start with some notation. We use $P = \{p_1, p_2, \ldots, p_n\}$ and $C = \{C_1, C_2, \ldots, C_m\}$ to denote the given points and circles, respectively. We denote with $c_1, c_2, \ldots, c_m$ and $r_1, r_2, \ldots, r_m$ the center points and radii of the m given circles, respectively.

In addition to n and m, which represent the number of points and circles, respectively, we consider the following parameters.

1. k: size of point-circle incidence graph $\mathcal{G}$: consider the bipartite graph $\mathcal{G}$ in which there is one vertex for each point in P and circle in C, and an edge (i, j) if and only if p_i is contained in circle C_j.
2. ρ: complexity of the arrangement $\mathcal{A}$ formed by the circles in C.
3. Δ: ply (i.e., depth) of $\mathcal{A}$.
4. $UNIT$: whether all circles are unit-size.
5. $GRID$: whether all centers lie on a square grid.
6. λ: ratio between circle radius and grid resolution (for case UNIT and GRID).

2.1 Relations between parameters

It is interesting to observe how the different parameters identified relate to each other, and how they affect the problem's complexity. Due to space constraints, here we only state several relations between the parameters, and defer a detailed analysis to the full version.

- Bounded ply implies bounded arrangement complexity: $\rho \in O(m\Delta)$.
- For unit circles, bounded arrangement complexity implies bounded ply: $UNIT \implies \Delta \in O(\sqrt{\rho})$.
- Bounded ply implies bounded complexity of the point-circle incidence graph: $k \in O(n\Delta)$.
- For grid circles, a unit radius implies bounded ply: $UNIT \wedge GRID \implies \Delta \in O(\lambda^2)$.

3 Algorithms

For a point p_i and circle C_j with center c_j and radius r_j, let the distance between p_i and c_j be $d_{ij} = ||p_i - c_j||$. Observe that if point p_i is inside C_j, then the maximum perturbation threshold for p_i and C_j in Problem 1 is $r_j - d_{ij}$: if the threshold is lower than this value, p_i is

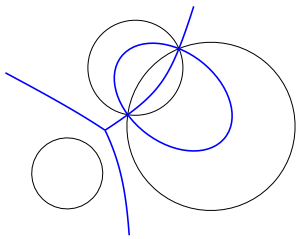


Figure 2: An example showing the Voronoi diagram of three circles (in blue). Note that every intersection between two circles is a vertex in this diagram.

guaranteed to remain inside C_j, if not, then p_i could be moved out of C_j. Similarly, if p_i is outside C_j, the critical threshold is $d_{ij} - r_j$. Note that $|d_{ij} - r_j|$ equals the distance from p_i to the closest point on C_j.

3.1 Brute-force approach

For Problem 1, the maximum perturbation threshold r is the minimum value of $|d_{ij} - r_j|$ $(1 \leq i \leq n, 1 \leq j \leq m)$. A brute-force solution to the problem implies computing all distances between points and circles, and requires $\Theta(nm)$ time.

For Problem 2, we look for the maximum threshold so that the number of points in each circle changes by at most a factor δ. This is easy to find by first computing how many points lie inside each circle, giving a threshold t_j for how many points may be moved into or out of C_j. Then we are looking for the t_jth closest points to C_j on the inside and outside. Both can be found in $O(n)$ time with a standard selection algorithm; doing this for all circles takes $O(nm)$ time.

Theorem 1 *Problems 1 and 2 can both be solved in $O(nm)$ time.*

3.2 Voronoi-based approach

The solution of Problem 1 is equivalent to finding the minimum point-circle distance. Therefore it is natural to use Voronoi diagrams to find this value. The Voronoi diagram of the circles C, which subdivides the plane into regions in which the closest circle boundary point is the same (see Fig. 2), can be used to easily find the minimum point-circle distance. The complexity of the Voronoi diagram of m circles is $O(\rho + m)$, and it can be computed in $O((\rho + m) \log m)$ time. Note that this Voronoi diagram is not the additively weighted Voronoi diagram—that has $O(m)$ size—since in that one the distance between a circle and a point inside is negative, but in our setting it is always positive.

Once the Voronoi diagram of C is computed, we can find the closest circle to each point by using standard point location data structures. For instance, a *layered dag* [5] can be built using $O(\rho+m)$ time and space, and allows point location queries in $O(\log(\rho + m))$ time. This leads to a total running time of $O((\rho + n) \log m)$. Note that, in the worst case, this is $\Theta((m^2 + n) \log m)$.

Theorem 2 *Problem 1 can be solved in $O((\rho + n) \log m)$ time.*

3.3 Counting-based approach

The Voronoi-based approach does not directly work for Problem 2, because it requires first to know how many points lie in each circle. In fact, the bottleneck in the general case is counting the number of points inside each circle. However, we can leverage well-known linearization techniques to speed this up.

3.3.1 Counting problem

We first consider the following basic question: given n points and m circles, determine the number of points inside each circle.

We can answer this question by transforming the points and circles in $\mathbb{R}^2$ to points and halfspaces in $\mathbb{R}^3$, using a standard lifting map (i.e. $(x, y) \mapsto (x, y, x^2+y^2)$). Then we can use a range searching data structure to improve the running time. In particular, Agarwal et al. [2] show that a set of n points can be preprocessed into a data structure of size s such that a circle counting query can be resolved in $O(\frac{n^{4/3}}{s^{2/3}} \log(s/n))$ time. If we want to do m such queries, then it is best to choose $s = m^{3/5}n^{4/5}$. Therefore the total running time for m queries becomes $O(n^{4/5}m^{3/5} \log \frac{m^3}{n})$, which is a clear improvement over the quadratic brute-force algorithm.

We can then approximate the computation to any desired precision using a standard binary search, or solve it exactly with parametric search, as shown next.

3.3.2 Parametric Search

It is possible to solve Problems 1 and 2 using the result in the previous section combined with parametric search [8]. Clearly, if we had $O(m)$ processors and a parallel algorithm that runs a single query on each processor, we could count the numbers of points in m circles in $O(\frac{n^{4/5}}{m^{2/5}} \log m)$ time. Plugging this into the parametric search framework we obtain the following result (details omitted for brevity).

Theorem 3 *Problems 1 and 2 can be solved in $O(m \log m + n^{8/5}m^{1/5} \log^3 m)$ time.*

3.3.3 Approach for unit circles

For unit circles one can do better. To begin with, note that in this case, the role of circles and points can be exchanged (obtaining a symmetric problem), thus we can assume that $n < m$. Firstly, we can avoid parametric search, by searching implicitly in a

polynomial set of candidate distances. We observe that all point-circle distances, in this case, are of the form $|pc| - 1$ or $1 - |pc|$, for some point p and some circle center c. We treat both forms separately, and for each form perform a binary search in the set of all point-to-point distances in $P \cup C$. Chan [4] shows how to select the kth largest point-to-point distance in a set of n points in $O(m^{4/3} \log^{5/3} m)$ expected time. Combining this with the algorithm in Section 3.3.1, we obtain an expected running time of $O(m^{4/3} \log^{8/3} m + n^{4/5} m^{3/5} \log^2 m)$ for both problems.

We can further improve this by using the fact that intersections between circles can be counted faster than point-halfspace incidences. Agarwal *et al.* show how to count the number of red-blue intersections in a set of unit circles in $O^*(n^{4/3})$ time [3]. Again, we consider the minimum amount we can shrink and grow the circles separately. We transform each circle of radius $1 - \varepsilon$ (resp. $1 + \varepsilon$) into a blue circle of radius $\frac{1}{2} - \frac{1}{2}\varepsilon$ (resp. $\frac{1}{2} + \frac{1}{2}\varepsilon$), and each point into a red circle of radius $\frac{1}{2} - \frac{1}{2}\varepsilon$ (resp. $\frac{1}{2} + \frac{1}{2}\varepsilon$). Then counting point-circle incidences becomes equivalent to counting the number of red-blue circle intersections.

Theorem 4 *For unit circles, Problems 1 and 2 can be solved in $O^*((m+n)^{4/3})$ expected time.*

We note that the previous approach is similar to the one used by Agarwal et al. [1], although their goal is slightly different (the running time is the same, up to logarithmic factors).

4 Discussion

We have presented several algorithms, ranging from brute-force to some that use advanced data structures, to solve Problems 1 and 2. Although it is unlikely that the general problem can be solved faster than $O((n+m)^{4/3})$ time, we have shown that in certain cases this can be achieved. The choice of which algorithm to use, in practice, depends on multiple factors, including the values of the parameters identified in Section 2. For instance, if the arrangement of circles has small complexity, the Voronoi-based approach should be a fast and practical way to solve Problem 1. On the other hand, if the size of the point-circle incidence graph is small, directly counting the points in each circle gives a fast way to solve both problems.

The most relevant particular case is probably that of UNIT and GRID, since this is exactly the setting from KDE. For this case, the fact that circles are unit-size allows us to use the faster algorithm in Theorem 4. Interestingly, we were not able to exploit the fact that the circles lie on a grid, which leads us to the following open problem.

Open Problem 1 *Given n points in $\mathbb{R}^2$, and a set of m unit-size circles, centered on a $\sqrt{m}$ by $\sqrt{m}$ grid, is it possible to count the number of points inside each circle in nearly-linear time?*

Acknowledgments M.L. was supported by the Netherlands Organisation for Scientific Research (NWO) under project no. 614.001.504. R.S. was partially supported by projects MINECO MTM2015-63791-R, Gen. Cat. DGR 2014SGR46, and by MINECO's Ramón y Cajal program.

References

[1] P. K. Agarwal, B. Aronov, M. Sharir, and S. Suri. Selecting distances in the plane. *Algorithmica*, 9(5):495–514, 1993.

[2] P. K. Agarwal and J. Matousek. On range searching with semialgebraic sets. *Discrete Comput. Geom*, 11:393–418, 1994.

[3] P. K. Agarwal, M. Pellegrini, and M. Sharir. Counting circular arc intersections. *SIAM J. Comput.*, 22(4):778–793, 1993.

[4] T. M. Chan. On enumerating and selecting distances. *Int. J. Comput. Geom. Appl.*, 291(11), 2001.

[5] H. Edelsbrunner, L. J. Guibas, and J. Stolfi. Optimal point location in a monotone subdivision. *SIAM J. Comput.*, 15(2):317–340, 1986.

[6] M. Kwan, I. Casas, and B. C. Schmitz. Protection of geoprivacy and accuracy of spatial information: How effective are geographical masks? *Cartographica*, 39(2):15–28, 2004.

[7] J. Luo, J. Liu, and L. Xiong. Privacy preserving publication of locations based on delaunay triangulation. In *Proc. PAKDD 2014*, pages 594–605, 2014.

[8] N. Megiddo. Applying parallel computation algorithms in the design of serial algorithms. *Journal of the ACM*, 30(4):852–865, 1983.

[9] V. Rastogi, D. Suciu, and S. Hong. The boundary between privacy and utility in data publishing. In *Proceedings of the 33rd International Conference on Very Large Data Bases*, VLDB '07, pages 531–542. VLDB Endowment, 2007.

[10] S. Shekhar, P. Zhang, Y. Huang, and R. R. Vatsavai. *Data Mining: Next Generation Challenges and Future Directions*, chapter Trends in Spatial Data Mining. AAAI/MIT Press, 2004.

[11] X. Shi, J. Alford-Teaster, and T. Onega. Kernel density estimation with geographically masked points. In *Proc. Geoinformatics 2009*, pages 1–4, 2009.

[12] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman & Hall/CRC, April 1986.

[13] X. Xiao, Y. Tao, and M. Chen. Optimal random perturbation at multiple privacy levels. *Proc. VLDB Endow.*, 2(1):814–825, Aug. 2009.

[14] P. Zandbergen. Ensuring confidentiality of geocoded health data: Assessing geographic masking strategies for individual-level data. *Advances in Medicine*, 2014(567049), 2014.

Bounding a global red-blue proportion using local conditions

Márton Naszódi *[1], Leonardo Martínez-Sandoval †[2], and Shakhar Smorodinsky‡[3]

[1]Department of Geometry, Lorand Eötvös University, Budapest, Hungary
[1]EPFL, Lausanne, Switzerland
[2]Department of Computer Science, Ben-Gurion University of the Negev, Be'er-Sheva Israel.
[3]Department of Mathematics, Ben-Gurion University of the Negev, Be'er-Sheva Israel.

Abstract

We study the following local-to-global phenomenon: Let B and R be two finite sets of (blue and red) points in the Euclidean plane $\mathbb{R}^2$. Suppose that in each "neighborhood" of a red point, the number of blue points is at least as large as the number of red points. We show that in this case the total number of blue points is at least one fifth of the total number of red points. We also show that this bound is optimal and we generalize the result to arbitrary dimension and arbitrary norm using results from Minkowski arrangements.

1 Introduction

Consider the following scenario in wireless networks. Suppose we have n clients and m antennas where both are represented as points in the plane (see Figure 1). Each client has a wireless device that can communicate with the antennas. Assume also that each client is associated with some disk centered at the client's location and having radius representing how far in the plane his device can communicate. Suppose also, that some communication protocol requires that in each of the clients disks, the number of antennas is at least some fixed proportion $\lambda > 0$ of the number of clients in the disk. Our question is: does such a local requirement imply a global lower bound on the number of antennas in terms of the number of clients? In this paper we answer this question and provide exact bounds. Let us formulate the problem more precisely.

Let B and $R = \{p_1, \dots, p_n\}$ be two finite sets in $\mathbb{R}^2$. Let $\mathcal{D} = \{D_1, \dots, D_n\}$ be a set of Euclidean disks centered at the red points, i.e., the center of D_i is p_i. Let $\{\rho_1, \dots, \rho_n\}$ be the radii of the disks in $\mathcal{D}$.

Theorem 1 *Assume that for each i we have $|D_i \cap B| \geq |D_i \cap R|$. Then $|B| \geq \frac{n}{5}$. Furthermore, the multiplicative constant $\frac{1}{5}$ cannot be improved.*

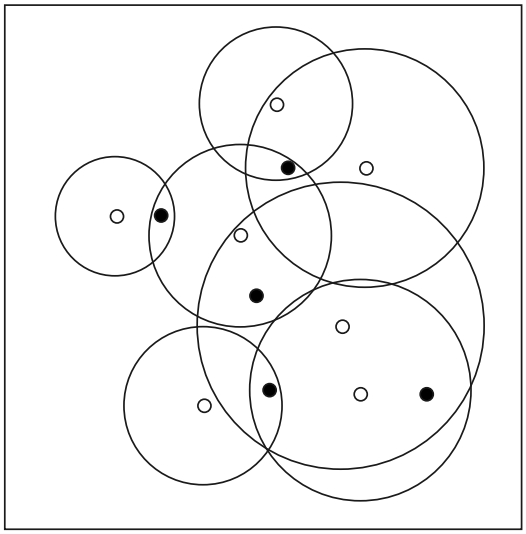


Figure 1: In each device range (each disk) there are at least as many antennas (black dots) as devices (white dots), so the hypothesis holds for $\lambda = 1$.

Such a local-to-global ratio phenomenon was shown to be useful in a more combinatorial setting. Pach et. al. [PRT15], solved a conjecture by Richter and Thomassen [RT95] on the number of total "crossings" that a family of pairwise intersecting curves in the plane in general position can have. Lemma 1 from their paper is a first step in the proof and it consists of a local-to-global phenomenon as described above.

We will obtain Theorem 1 from a more general result. In order to state it, we introduce some terminology.

Let K be an origin-symmetric convex body in $\mathbb{R}^d$, that is, the unit ball of a norm.

A *strict Minkowski arrangement* is a family $\mathcal{D} = \{K_1 = p_1 + \rho_1 K, \dots, K_n = p_n + \rho_n K\}$ of homothets of K, where $p_i \in \mathbb{R}^d$ and $\rho_i > 0$, such that no member of the family contains the center of another member. An *intersecting family* is a family of sets that all share at least one element.

We denote the *maximum cardinality of an intersecting strict Minkowski arrangement* of homothets of K

*Email address: marton.naszodi@math.elte.hu.
†Email address: leomtz@im.unam.mx.
‡Email address: shakhar@math.bgu.ac.il.

by $M(K)$. It is known that $M(K)$ exists for every K and $M(K) \leq 3^d$ (see, e.g., Lemma 21 of [NPS16]). On the other hand (somewhat surprisingly), there is an origin-symmetric convex body K in $\mathbb{R}^d$ such that $M(K) = \Omega\left(\sqrt{7}^d\right)$, [Tal98, NPS16]. For more on Minkowski arrangements see, e.g., [FL94].

We need the following auxiliary Lemma.

Lemma 2 *Let K be an origin-symmetric convex body in $\mathbb{R}^d$. Let $R = \{p_1, \ldots, p_n\}$ be a set of points in $\mathbb{R}^d$ and let $\mathcal{D} = \{K_1 = p_1 + \rho_1 K, \ldots, K_n = p_n + \rho_n K\}$ be a family of homothets of K. Then there exists a subfamily $\mathcal{D}' \subset \mathcal{D}$ that covers R and forms a strict Minkowski arrangement. Moreover, $\mathcal{D}'$ can be found using a greedy algorithm.*

As a corollary, we will obtain the following theorem.

Theorem 3 *Let K be an origin-symmetric convex body in $\mathbb{R}^d$. Let $R = \{p_1, \ldots, p_n\}$ be a set of points in $\mathbb{R}^d$ and let $\mathcal{D} = \{K_1 = p_1 + \rho_1 K, \ldots, K_n = p_n + \rho_n K\}$ be a family of homothets of K where $\rho_1, \ldots, \rho_n > 0$. Let B be another set of points in $\mathbb{R}^d$, and assume that, for some $\lambda > 0$, we have*

$$\frac{|B \cap K_i|}{|R \cap K_i|} \geq \lambda, \tag{1}$$

for all $i \in [n]$. Then $\frac{|B|}{|R|} \geq \frac{\lambda}{3^d}$.

In Theorem 1 the convex body K is a Euclidean unit disk in the plane. Another case of special interest is when the convex body K is a unit cube and thus it induces the ℓ_∞ norm. In this situation we get a sharper and optimal inequality.

Theorem 4 *If K is the unit cube in $\mathbb{R}^d$, then the conclusion in Theorem 3 can be strengthened to $\frac{|B|}{|R|} \geq \frac{\lambda}{2^d}$. Furthermore, the multiplicative constant $\frac{1}{2^d}$ cannot be improved.*

In the results above, the points p_i play the role of the centers of the sets of the Minkowski arrangement. One might ask if this restriction is essential. As a final result, we give a general construction to show that it is.

Theorem 5 *Let K be any convex body in the plane and ε, λ any positive real numbers. There exist sets of points $R = \{p_1, \ldots, p_n\}$ and B in the plane such that $|B| < \varepsilon n$ and that for each i there is a translate K_i of K that contains p_i for which $|B \cap K_i| \geq \lambda |R \cap K_i|$.*

In particular, even if each red point is contained in a unit disk with many blue points, the global blue to red ratio can be as small as desired. This is a possibly counter-intuitive fact in view of Theorem 1.

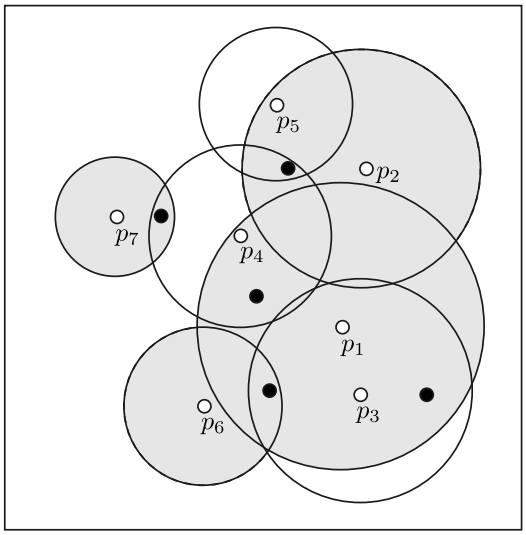


Figure 2: The centers of the disks are labeled in decreasing order of corresponding radii. The shaded disks cover the white points and no shaded disk contains the center of another.

2 Proofs

Proof. [Proof of Lemma 2] We construct a subfamily $\mathcal{D}'$ of $\mathcal{D}$ with the property that no member of $\mathcal{D}'$ contains the center of any member of $\mathcal{D}'$, and $\bigcup \mathcal{D}'$ covers the red points, R. Assume without loss of generality that the labels of the points in R are sorted in non-increasing order of the homothety ratio, that is, $\rho_1 \geq \cdots \geq \rho_n$. See Figure 2 for an example.

We construct $\mathcal{D}'$ in a greedy manner as follows: Add K_1 to $\mathcal{D}'$. Among all red points that are not already covered by $\mathcal{D}'$ pick a point p_j whose corresponding homothet K_j has maximum homothety ratio ρ_j. Add K_j to $\mathcal{D}'$ and repeat until all red points are covered by $\mathcal{D}'$. Note that the homothets in $\mathcal{D}'$ are not necessarily disjoint.

Clearly, $R \subset \bigcup \mathcal{D}'$. Now we show that no member of $\mathcal{D}'$ contains the center of another. Suppose to the contrary that K_i contains the center of K_j. If $i < j$, then $\rho_i \geq \rho_j$ so K_i was chosen first, a contradiction to the fact that p_j was chosen among the points not covered by previous homothets. If $i > j$, then K_j also contains the center of K_i, and we get a similar contradiction.

This finishes the proof of Lemma 2.

□

Proof. [Proof of Theorem 3] By Lemma 2, there exists a subfamily $\mathcal{D}' \subset \mathcal{D}$ that covers R and form a strict Minkowski arrangement. Namely, $\bigcup \mathcal{D}'$ covers R, and no point of B is contained in more than $M(K)$ members of $\mathcal{D}'$. In particular, it follows that

$$|R| \leq \sum_{K \in \mathcal{D}'} |R \cap K| \leq \sum_{K \in \mathcal{D}'} \frac{|B \cap K|}{\lambda} \leq \frac{M(k)}{\lambda} |B|$$

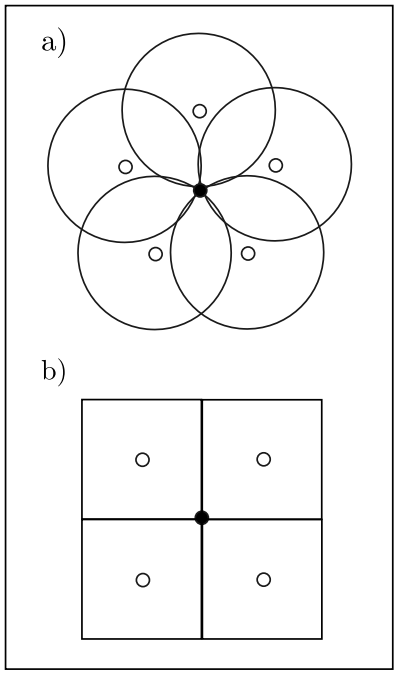


Figure 3: Optimal Minkowski arrangements in the plane for a) Euclidean disks, b) axis-parallel squares.

so

$$\frac{|B|}{|R|} \geq \frac{\lambda}{M(K)} \geq \frac{\lambda}{3^d}.$$

This completes the proof. □

Lemma 6 *Let K be the Euclidean unit disk centered at the origin. Then $M(K) = 5$.*

Proof. [Proof of Lemma 6] Five unit disks centered in the vertices of a unit-radius regular pentagon show that $M(K) \geq 5$. See Figure 3a.

To prove the other direction, suppose that there is a point b in the plane that is contained in 6 Euclidean disks in a strict Minkowski arrangement. Then, by the pigeonhole principle, there are two centers of those disks, say p and q such that the angle $\sphericalangle(pbq)$ is at most $60°$. Assume without loss of generality that $pb \geq qb$. It is easily verified e.g., by the law of cosines, that the distance pq is less than pb. Hence, the disk centered at p contains q, a contradiction. This completes the proof. □

Lemma 7 *Let K be the unit cube of $\mathbb{R}^d$ centered at the origin. Then $M(K) = 2^d$.*

Proof. [Proof of Lemma 7] Let d be a positive integer and $e_1, e_2, \ldots, e_n$ the canonical base of $\mathbb{R}^d$. Consider all the cubes of radius 1 centered at each point of the form $\pm e_1 \pm e_2 \pm \ldots \pm e_d$. This family shows that $M(K) \geq 2^d$. See Figure 3b for an example on the plane.

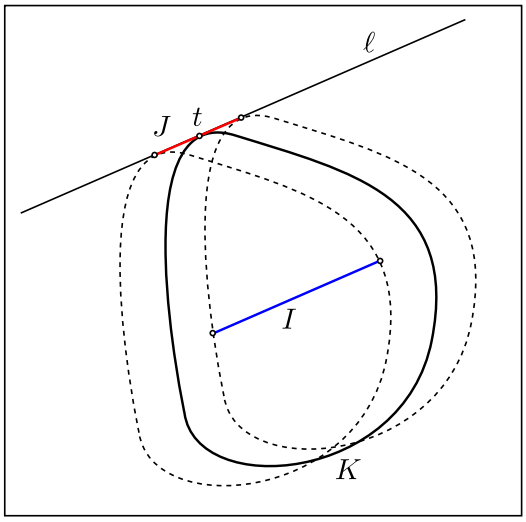


Figure 4: Construction of example without local-to-global phenomenon.

Now we show the other direction. Consider the 2^d closed regions of $\mathbb{R}^d$ bounded by the hyperplanes $x_i = 0$ $i = 1, 2, \ldots, d$ and suppose on the contrary that we have an example with $2^d + 1$ cubes or more that contain the origin. By the pidgeon-hole principle there is a region with at least two cube centers u and v. By applying a rotation we may assume that it is the region of vectors with non-negative entries. We may also assume $\delta := \|u\|_\infty \geq \|v\|_\infty$.

Since the d-cube centered at u contains the origin, its radius must be at least δ. We claim that this cube contains v. Indeed, each of the entries of u and v are in the interval $[0, \delta]$. So each of the entries of $u - v$ are in $[-\delta, \delta]$. Then $\|u - v\|_\infty \leq \delta$ as claimed. This contradiction finishes the proof. □

Theorem 1 clearly follows from combining the proof of Theorem 3 (with $\lambda = 1$) and Lemma 6. The result is sharp because we have equality when R is the set of vertices of a regular pentagon with center p and $B = \{p\}$. Similarly, Theorem 4 and its optimality follow from Lemma 7.

Remark 1 *Lemma 6 can be generalized to arbitrary dimension. This implies that Theorem 1 can be generalized to arbitrary dimension almost verbatim.*

Proof. [Proof of Theorem 5] Let K be any convex body in the plane. We construct sets R and B as follows. Let ℓ be a tangent line of K which intersects K at exactly one point t. Let I be a non-degenerate closed line segment contained in K and parallel to ℓ. Let J be the (closed) segment that is the locus of the point t as K varies through all its translations in direction d that contain I. See Figure 4.

We construct R by taking any n points from J and we construct B by taking any m points from I. For

any point in R there is a translation of K that contains exactly one point of R and m points of B, which makes the local B to R ratio equal to m. But globally we can make the ratio $\frac{m}{n}$ arbitrarily small. □

Acknowledgements

M. Naszódi acknowledges the support of the János Bolyai Research Scholarship of the Hungarian Academy of Sciences, and the National Research, Development, and Innovation Office, NKFIH Grant PD-104744, as well as the support of the Swiss National Science Foundation grants 200020-144531 and 200020-162884.

L. Martinez-Sandoval's research was partially carried out during the author's visit at EPFL. The project leading to this application has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme grant No. 678765 and from the Israel Science Foundation grant No. 1452/15.

S. Smorodinsky's research was partially supported by Grant 635/16 from the Israel Science Foundation. A part of this research was carried out during the author's visit at EPFL, supported by Swiss National Science Foundation grants 200020-162884 and 200021-165977.

References

[FL94] Z. Füredi and P. A. Loeb, *On the best constant for the Besicovitch covering theorem*, Proc. Amer. Math. Soc. **121** (1994), no. 4, 1063–1073. MR1249875 (95b:28003)

[NPS16] M. Naszódi, J. Pach, and K. Swanepoel, *Arrangements of homothets of a convex body*, arXiv:1608.04639 [math] (2016). arXiv: 1608.04639.

[PRT15] J. Pach, N. Rubin, and G. Tardos, *Beyond the Richter-Thomassen conjecture*, arXiv:1504.08250 [math] (2015). arXiv: 1504.08250.

[RT95] R. B. Richter and C. Thomassen, *Intersections of curve systems and the crossing number of $C_5 \times C_5$*, Discrete & Computational Geometry **13** (1995), no. 2, 149–159.

[Tal98] I. Talata, *Exponential lower bound for the translative kissing numbers of d-dimensional convex bodies*, Discrete Comput. Geom. **19** (1998), no. 3, Special Issue, 447–455. Dedicated to the memory of Paul Erdős. MR98k:52046

Convex allowable sequences*

Jean Cardinal Udo Hoffmann

Abstract

Allowable sequences are a well known abstraction of combinatorial properties of a planar point set. We consider the allowable sequences that describe a weakly convex point set and show that the *realizability problem* for weakly convex allowable sequences of a point set on three lines is complete in the *existential theory of the reals* ($\exists\mathbb{R}$), while the realizability of weakly convex allowable sequences on two lines can be decided in polynomial time. Furthermore, we count the simple convex allowable sequences.

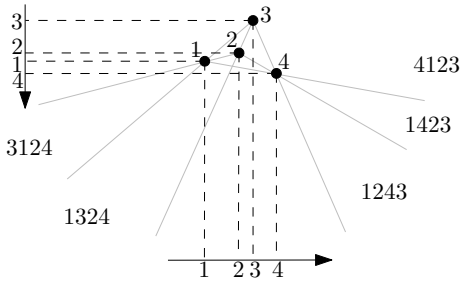


Figure 1: An example of an allowable sequence of a point set.

1 Introduction

Combinatorial abstractions of properties of point sets are a useful tool in computational geometry. For example, many algorithms on point sets only require the *order type* instead of the exact coordinates of the points, that is, the orientation (clockwise/counterclockwise) of every triple of points in the set, which makes the algorithms purely combinatorial. So it is a natural question to ask how the point sets that agree on their combinatorial description – the *realization space* – looks like. For order types this question was answered by Mnëv [11] with his famous universality theorem: For each *primary semi-algebraic* set S (a set described by strict polynomial inequalities and polynomial equalities) there is an order type whose realization space is *stably equivalent* to S. So the realization space can be as complex as possible. From the computational point of view Mnëv's universality theorem implies that deciding if an order type is *realizable* (i.e., the realization space is non-empty) is complete in the *existential theory of the reals* ($\exists\mathbb{R}$).

A sequence $\pi_1, \pi_2, \ldots, \pi_{m+1}$ of permutations of an n-element set is an *allowable sequence* if: (i) π_1 and π_{m+1} are reverse of each other, (ii) every pair of elements is reversed exactly once, (iii) each consecutive pair of permutations differ only by reversals of disjoint substrings. We call an allowable sequence *simple* if each consecutive pair of permutations only differs by the reversal of a single adjacent transposition. (Note that in what follows, we think of permutations as strings in the alphabet $[n]$.)

A simple allowable sequence can be obtained from a point set in general position in the following way. The orthogonal projection of the point set onto an oriented line leads to a permutation. Rotating this line by a continuous motion by 180° leads to a sequence of permutations which is an allowable sequence. The allowable sequence corresponding to a point set encodes in particular the information of the order type of the point set [6]. The concept of simple allowable sequence has first been described by Perrin [12] in 1882 who conjectured that every such sequence is realizable by a set of points. This was disproved by Goodman and Pollack [6] with the so-called "bad pentagon" construction. The authors generalized the definition in [7] to what we call allowable sequence in this paper.

Each of the reversals between two successive permutations will be referred to as a *switch*. The elements of one substring s that is reversed correspond to points that lie on one line ℓ_s. This substring is reversed when the rotating line, which we project on to obtain the permutations, is orthogonal to ℓ_s. We can identify a switch with the intersection point of the line spanned by its points and the line at infinity ℓ_∞ and a single permutation with the interval between two intersection points on the line at infinity.

The order of switches corresponds exactly to the order of the slopes of the spanned lines. In an allowable sequence we can have *parallel switches*, that is, more than one reversal occuring between two successive permutations. If this appears in an allowable sequence corresponding to a point set, then the parallel switches correspond to parallel lines containing at least two points each.

We consider the *realizability problem* for allowable sequences: given such a sequence, can it be obtained from a set of points in the plane? We further restrict our investigations to allowable sequences that

*Université libre de Bruxelles (ULB), [udo.hoffmann,jcardin]@ulb.ac.be

can only be obtained by sets of points in convex position. (Note that n-point sets in convex position all have the same order type, but may have different allowable sequences.) The realizability problem for (non-convex) allowable sequences has been shown to be $\exists\mathbb{R}$-complete [8]. We say that a simple allowable sequence is *convex* whenever every element in $[n]$ appears either as the first or the last in one permutation of the sequence. The same definition holds for general allowable sequences. Furthermore, an allowable sequence is said to be *weakly convex* whenever every element in $[n]$ is involved in a switch of a prefix or a suffix of one of the permutations in the sequence. Such sequences can be produced by point sets such that all the points lie on the boundary of their convex hull, but are not vertices of the convex hull.

We also consider allowable sequences that are produced by points lying on a few lines. An allowable sequence is said to be *on k lines* if it involves k switches, such that every element is involved in at least one of them. Such a sequence can only be obtained by point sets that can be covered by k lines.

The *existential theory of the reals* ($\exists\mathbb{R}$) is a complexity class defined by the following complete problem: given a Boolean combination of polynomial equalities and inequalities, decide if there is an assignment of real values to the variables such that the system is satisfied. The only known relations to other complexity classes are $NP \subseteq \exists\mathbb{R} \subseteq PSPACE$. The second relation is a result by Canny [3]. Whether one of the relations is strict or an equality is not known. An argument suggesting that $\exists\mathbb{R}$ may not be contained in NP is the fact that the normal certificate for $\exists\mathbb{R}$ hard problems, the coordinates of a solution, cannot always be stored in polynomial space using the standard bit representation for integers since iterative squaring produces doubly exponential numbers in the input size, which requires exponential size bit representations of numbers. Many geometric problems can be shown to be $\exists\mathbb{R}$-hard by a reduction from order type realizability or the dual problem, the stretchability of pseudoline arrangements. Some $\exists\mathbb{R}$-complete geometric problems include recognition of segment [10] and convex set intersection graphs [14] and point visibility graphs [5], realizability of face lattice of a 4-polytope [13] and d-dimensional Delaunay triangulations [1], computing the rectilinear crossing number [2] and planar slope number [9]. We refer to [4] for an overview.

Results and outline

In Section 2 we first show that the realizability problem for weakly convex allowable sequences is $\exists\mathbb{R}$-complete. The points in the reduction are placed on three lines. We show that three lines are necessary in the reduction by showing that the realizability of weakly convex allowable sequences on two lines can be decided in polynomial time, even though the coordinates of a realization sometimes require exponential space bit representations. In Section 3 we count the number of simple convex allowable sequences.

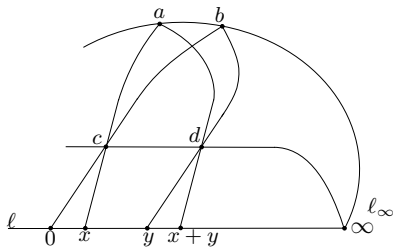


Figure 2: The addition of distances on a line.

2 Realizability

2.1 Weakly convex position

Theorem 1 *The realizability problem for weakly convex allowable sequences on three lines is $\exists\mathbb{R}$-complete.*

We leave the $\exists\mathbb{R}$-membership of the problem for the reader. The idea to prove the $\exists\mathbb{R}$-hardness is to calculate with points on a line as in the proof of Mnëv's universality theorem. Therefore, we use the standard gadgets of von Staudt [16] to add and multiply distances on a line.

Figure 2 shows the addition of distances on a line. The distance between 0 and y is the same as the distance between c and d, because of the parallel lines that intersect on the line at infinity in b. This distance is again the same distance as between x and $x + y$ and thus this point configuration adds the distances of points on ℓ if ℓ_∞ is indeed the line at infinity. Since distances are not invariant under projective transformations we use a projective invariant, the *cross-ratio*.

The cross-ratio $(a, b; c, d)$ of four points $a, b, c, d \in \mathbb{R}^2$ is defined as $(a, b; c, d) := \frac{|a,c|\cdot|b,d|}{|a,d|\cdot|b,c|}$, where $|x, y|$ is the determinant of the matrix obtained by writing the coordinates of the two points as columns. The two properties that are useful for our purpose are that the cross-ratio is invariant under projective transformations, and that for four points on one line, the cross-ratio is given by $\frac{\overrightarrow{ac}\cdot\overrightarrow{bd}}{\overrightarrow{ad}\cdot\overrightarrow{bc}}$, where $\overrightarrow{xy}$ denotes the oriented distance between x and y on the line.

To encode a complete semi-algebraic set with the gadgets we have to solve two problems. We have to break down the description of the semi-algebraic set to basic additions and multiplications and we need the order of the variables on the line ℓ. A solution to this problem is the following theorem.

Theorem 2 (Shor normal form [15]) *The following decision problem is $\exists\mathbb{R}$-complete: given variables $1 = x_1 < x_2 < \cdots < x_n$ and equations of the form $x_i + x_j = x_k$ and $x_i + x_j = x_k$, is there a satisfying assignment of real values to the variables?*

To implement those calculations in an allowable sequence we note that we can encode points on one line in the allowable sequence. Thus we move the line ℓ to the line at infinity as shown in Figure 3, which allows us to calculate with slopes of the spanned line.

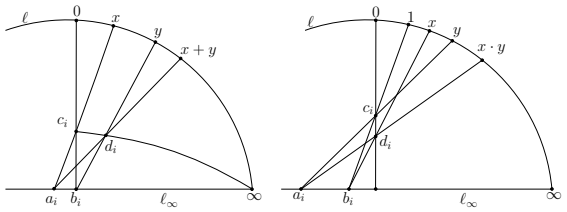


Figure 3: The gadgets calculating with distances on the line at infinity.

To prove the $\exists\mathbb{R}$-hardness of the realizability of convex allowable sequences we construct an convex allowable sequence A_S from a description of a semi-algebraic set S given in Shor normal form. We give a construction of A_S iteratively by placing the points of the gadgets. Therefore, let $m_1, \ldots, m_k$ be the multiplication gadgets and $a_1, \ldots, a_l$ be the addition gadgets. We add first multiplication gadgets and then the addition gadgets and construct the allowable sequence of the gadget points already placed. The points a_i and b_i are placed on the line ℓ_∞ given by $y = 0$, the points c_i and d_i of a multiplication gadget are placed on a line B and the points c_i and d_i of an addition gadget are placed on a line parallel to the line containing c_i and d_i.

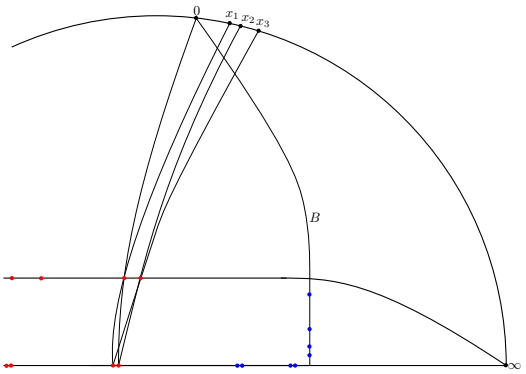


Figure 4: The placement of the gadget points in the reduction.

So, when we add the multiplication gadget m_i, we have to determine how the slopes of the lines spanned by pairs of points including a point of m_i fit in the already constructed sequence. We do this in the following way: The slopes spanned by the points a_i and b_i and the points on B of the gadgets m_j for $j < i$ are the smallest positive slopes already placed so far. Furthermore, for each point p the slopes of the lines $\ell(a_i, p)$ and $\ell(b_i, p)$ appear consecutively in the allowable sequence. These properties can be achieved when realizing the allowable sequence by moving a_i and b_i very close together and very far to the left on the line ℓ_∞. The slopes of lines through the points c_i and d_i are the largest positive slopes constructed so far. Let $(a_i, b_i,)a_{i-1}, b_{i-1}, \ldots, a_1, b_1$ be the order of the gadget points on ℓ_∞. Then the order of slopes of lines through c_i and d_i in increasing order is $(a_{i-1}, d_i), (b_{i-1}, d_i), (a_{i-1}, c_i), (b_{i-1}, c_i), (a_{i-2}, d_i), (b_{i-2}, d_i), (a_{i-2}, c_i), (b_{i-2}, c_i), \ldots$.

The new order of the slopes after we add an addition gadget g_i is the following. The slopes of lines through the points a_i and b_i are again the smallest positive slopes and the slopes through c_i and d_i are the largest negative slopes (smallest absolute value).

Lemma 3 *The allowable sequence A_S as described above is realizable if and only if S is non-empty.*

Proof. [Sketch.] If A_V is realizable, then the gadgets implement relations given by V of the points on the line at infinity. Thus the cross-ratios of points on the line at infinity encodes a point in V.

On the other hand, if V is non-empty, let x be a point in V. We place the points on the line at infinity according to the cross-ratio. Then we place the gadget points in the order we constructed the allowable sequence on ℓ_∞. Here we place a_i and b_i close together and far to the left, which leads to the desired allowable sequence. □

To finish the proof of the theorem, note that the sequence A_v describes a point set on three lines.

2.2 Points on few lines

Note that the allowable sequence we described in the reduction describes a point set that lies on three sides of a trapezoid, i.e., on three lines. We show that the realizability of an allowable sequence by a point set on two lines can be decided in polynomial time.

Theorem 4 *The realizability of weakly convex allowable sequences on two lines can be decided in polynomial time.*

Proof. We distinguish two cases. In the first case we assume the points lie on two parallel lines. Here we can assume that the two lines are given by $x = 0$ and $x = 1$ by applying an affine transformation. The allowable sequence describes the order of slopes between pairs of points on the two lines. The slope through two points $a = (0, y_a)$ and $b = (1, y_b)$ is given

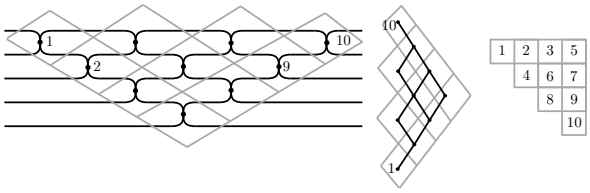


Figure 5: An order of intersections in the wiring diagram is in bijection with a linear extension of the partial ordered set and a skewed Young tableau of triangle shape.

by $y_b - y_a$. Consequently, the order of the slopes can be encoded by a linear inequality system. Since the solvability of a system of linear inequalities can be decided in polynomial time we can decide the realizability of allowable sequences on two parallel lines in polynomial time.

In the second case, when the two lines are not parallel, we can assume again using an affine transformation that the two lines supporting the points are given by $x = 0$ and $y = 0$ and all slopes are positive, e.g., all x-coordinates are non-positive and all y-coordinates are non-negative. The slope through a point $a = (0, y_a)$ and a point $b = (x_b, 0)$ is given by $-y_a/x_b$. Comparing slopes of pairs of lines leads to inequalities of the form $-y_a/x_b < -y_c/x_d$. Taking logarithms of each side of an (in)equality leads a linear inequality system where the variables correspond to logarithms of the realizing coordinates. This system can be solved in polynomial time. □

In the second case we do not compute the coordinates of the realizing points explicitly in polynomial time. This is not possible due to the following fact.

Observation 1 *There are realizable weakly convex allowable sequences on two lines, all realizations of which with points lying on the integer grid require at least one point with at least one coordinate of absolute value* $\Theta(2^{2^n})$.

Proof. [Sketch.] The multiplication gadgets used in the $\exists\mathbb{R}$-reduction can be realized only on two lines. Thus we can implement the reduction of Theorem 1 on the Shor normal form $0 < x_1 = 1 < x_2 \cdots < x_n$ with the equations $x_{i+1} = x_i \cdot x_i$ for $i > 2$. For $x_2 = p/q$ (p, q coprime) we obtain $x_n = p^{2^{n-2}}/q^{2^{n-2}}$ which requires doubly exponential resolution. □

3 Counting

Theorem 5 *The number of simple convex allowable sequences on n elements is* $\frac{\binom{n+1}{2}!(1!2!\ldots(n-1)!)}{(1!3!\ldots(2n-1)!)}$.

Proof. Figure 5 left shows the dual wiring diagram of a convex point set. An allowable sequence corresponds to the order of x-coordinates of intersection points in the wiring diagram. An order of the intersection points can be realized by a wiring diagram if the intersection points of each line appear in the correct order. This corresponds to a linear extension of the partial ordered set shown in the middle of Figure 5. Writing the position of an element in a linear extension in the grid indicated in grey leads to a *skewed Young tableau*, a filling of the shape in the right of the figure with numbers $\{1, \ldots, \binom{n}{2}\}$ that is strictly increasing in row and column. Which results in the given formula by [17]. □

References

[1] K. A. Adiprasito, A. Padrol, and L. Theran. Universality theorems for inscribed polytopes and Delaunay triangulations. *DCG*, 54:412–431, 2015.

[2] D. Bienstock. Some provably hard crossing number problems. *DCG*, 6:443–459, 1991.

[3] J. Canny. Some algebraic and geometric computations in PSPACE. In *STOC*, pages 460–467. ACM, 1988.

[4] J. Cardinal. Computational geometry column 62. *ACM SIGACT News*, 46:69–78, 2015.

[5] J. Cardinal and U. Hoffmann. Recognition and complexity of point visibility graphs. *DCG*, 57:164–178, 2017.

[6] J. E. Goodman and R. Pollack. On the combinatorial classification of nondegenerate configurations in the plane. *J. of Comb. Th., A*, 29:220–235, 1980.

[7] J. E. Goodman and R. Pollack. A theorem of ordered duality. *Geometriae Dedicata*, 12:63–74, 1982.

[8] U. Hoffmann. *Intersection Graphs and Geometric Objects in the Plane*. PhD thesis, TU Berlin, 2016.

[9] U. Hoffmann. On the complexity of the planar slope number problem. *JGAA*, 21:183–193, 2017.

[10] J. Kratochvíl and J. Matoušek. Intersection graphs of segments. *J. of Comb. Th., B*, 62:289–315, 1994.

[11] N. E. Mnëv. The universality theorems on the classification problem of configuration varieties and convex polytopes varieties. In *Top. a. Geom. – Rohlin Seminar*, LNM, pages 527–543. Springer, 1988.

[12] R. Perrin. Sur le problème des aspects. *Bulletin de la Société Mathématique de France*, 10:103–127, 1882.

[13] J. Richter-Gebert and G. M. Ziegler. Realization spaces of 4-polytopes are universal. *Bulletin of the American Mathematical Society*, 32:403–412, 1995.

[14] M. Schaefer. Complexity of some geometric and topological problems. In *GD*, volume 5849 of *LNCS*, pages 334–344. Springer, 2009.

[15] P. W. Shor. Stretchability of pseudolines is NP-hard. *Appl. Geo. Disc. Math: Victor Klee Festschrift*, 4:531–554, 1991.

[16] K. G. C. Staudt. *Geometrie der Lage*. F. Korn, 1847.

[17] R. M. Thrall et al. A combinatorial problem. *The Michigan Mathematical Journal*, 1(1):81–88, 1952.

Parametrized Runtimes for Ball Tournaments

Stefan Funke [i] Sabine Storandt [ii]

Abstract

The ball tournament problem asks for the elimination sequence of a given set of prioritized balls in $\mathbb{R}^d$, specified by their centers and their radii. The radii grow linearly over time, and whenever two balls touch, the one with the lower priority gets eliminated (and added to the elimination sequence), until only the ball with the highest priority remains. A simple algorithm for this problem has a running time of $\mathcal{O}(n^2 \log n)$. We present two parametrization variants for the ball tournament problem: An $\mathcal{O}(n \log \Delta(\log n + \Delta^{d-1}))$ algorithm, where Δ is the ratio of the largest to the smallest radius, and an $\mathcal{O}(Cn \log^{\mathcal{O}(1)} n)$ algorithm for $d = 2$ (assuming radii and priorities are positively correlated) where C denotes the number of different radii. Hence for $\Delta, C \in \mathcal{O}(\log^{\mathcal{O}(1)} n)$, we get a significantly improved running time of $\mathcal{O}(n \log^{\mathcal{O}(1)} n)$.

1 Introduction

Given a set of n balls B, where each ball $B = (c, r, p)$ has a center $c \in \mathbb{R}^d$, a radius $r \in \mathbb{R}^+$, and a priority value $p \in \mathbb{R}^+$, a *ball tournament* is modeled as follows: Time t, starting with $t = 0$, progresses continuously and the balls' radii grow linearly over time. For any time t, center c_i induces a ball with radius $r_i t$. If two balls touch/collide, the one with lower priority gets eliminated. The tournament ends when only one ball remains. The resulting sequence of the balls, sorted by their elimination time, is the *elimination sequence*.

A simple algorithm consists of computing all pairwise collision times of balls in B. These $\mathcal{O}(n^2)$ collision events are considered one-by-one in increasing time order. If the two balls involved in the collision are 'alive' at that time, the one with lower priority 'dies', otherwise the event is ignored. The running time of this algorithm is dominated by sorting the collision times and hence equals $\mathcal{O}(n^2 \log n)$.

A lower bound of $\Omega(n \log n)$ for the running time can be obtained by reduction from the closest pair problem. The scope of this paper is to show that we can get close to this lower bound in case certain parameters are small.

[i]Department of Computer Science, Universität Stuttgart, Germany, funke@fmi.uni-stuttgart.de

[ii]Department of Computer Science, JMU Würzburg, Germany, storandt@informatik.uni-wuerzburg.de

1.1 Related work

Another problem where the elimination sequence plays a role is the construction of motorcycle graphs (related to computing straight skeletons of polygons). Here, a set of n motorcycles, each with a start position, a direction and a given speed, continuously move in the plane until they 'crash' in the track left by any other motorcycle. The best running time known for motorcycle graph construction is $\mathcal{O}(n^{4/3+\epsilon})$ [4], $\mathcal{O}(Cn \log^2(n) \min(C, \log n))$ in case only C directions are allowed [4].

The ball tournament problem was considered for $d = 2$ in [2], proving a time bound of $O(n(\log^6 n + \Delta^2 \log^2 n + \Delta^4 \log n))$. The result was recently generalized and improved in [1] to $O(\Delta^d n(\log n + \Delta^d))$ for arbitrary d. In this paper we improve the dependency on Δ in all dimensions (e.g., for $d = 2$ from $O(\Delta^2 n \log n + \Delta^4 n)$ to $O(n \log \Delta \log n + \Delta n \log \Delta)$) and also introduce a new meaningful parametrization.

1.2 Nearest neighbor and range reporting queries

Our algorithms require efficient nearest neighbor (NN) and range reporting (RR) queries. In higher dimension, $d \geq 3$, one has to rely on approximate queries, as no exact data structure supporting polylogarithmic query times is known so far. So an ϵ-approximate NN (ANN) of q is a point $c \in P$ with $|qc| \leq (1+\epsilon)|qc'|$, for all $c' \in P$. An ϵ-approximate RR (ARR) query with range parameter r returns a set S with $S \supseteq \{c \in P : |qc| \leq r\}$ and $S \subseteq \{c \in P : |qc| \leq (1+\epsilon)r\}$. For arbitrary d, a so-called *quadtreap* [3] has an expected size of $\mathcal{O}(n \log n)$ with high probability. It supports ANN and ARR queries in expected $O(h + \left(\frac{1}{\epsilon}\right)^{d-1} + k)$ time where k is the output size ($k = 1$ for ANN), and point deletion and insertion in expected $O(\log n)$ time.

1.3 Collision and update events

Two different kinds of events are considered in our algorithm: *collision events* $(b, b', t_{col}(b, b'))$ (as in the naive algorithm) and *update events* (b, t) which trigger reexamination for collisions of b at time t. We call both events *anchored* at b.

In our algorithm, a collision event $(b, b', t_{col}(b, b'))$ will predict the potential collision of b with a *smaller* ball b' at time $t_{col}(b, b')$. An update event (b, t) is created if b certainly does not collide with a smaller

ball b' before time t. This can be ensured if b – as the larger of the two balls – contains at least half of the segment cc', i.e. $rt > |cc'|/2$. So if we know the NN distance l for c, setting $t = l/(2r)$ is a valid update time. As we only get the ANN distance $l^* \leq l(1+\epsilon)$, we use $t = l^*/(2(1+\epsilon)r)$ instead.

2 An algorithm with improved Δ-Dependency

We maintain a priority queue PQ of update and collision events, sorted by their respective event times increasingly. We now sketch the basic algorithm.

1. For all $b \in B$ create an ANN-based update event and insert it into the PQ.
2. As long as the PQ is not empty, extract the next event. Let b be the anchor of the event and t the event time. If b is dead, there is nothing to do. For b still alive, if it is

- an update event, compute the current ANN and the respective new update time t_{new} for b. If for t_{new} we have $t_{new} \leq (1+\epsilon)t$, predict the next collision event for b with a ball b' of smaller or equal radius and insert this event in the PQ. Otherwise, insert the update event (b, t_{new}).
- a collision event and both b and b' are alive, eliminate the one with lower priority and append it to the elimination sequence. If b survives, compute an update event for b. If the computed update time is before t, predict a collision event for b instead.
- a collision event and b' is dead, compute an update event for b. Again, if the update time is before t, predict a collision event instead.

In the following, we will analyze the running time.

2.1 Computing updates and predicting collisions

To have access to balls with smaller or equal radius, we first partition the n balls into $\log_2 \Delta$ classes according to their radii. So class number i contains balls with radii $r \in [2^i r_{min}, 2^{i+1} r_{min}[$ for $i = 0, \cdots, \log \Delta$. Then for each class, we construct a separate quadtreap. In total this takes time $\mathcal{O}(n \log n)$.

Lemma 1 *An update event can be computed in time* $\mathcal{O}(\log \Delta(\log n + 1/\epsilon^{d-1}))$.

Proof. For a ball b we issue an ANN query to the DS of every class with smaller or equal radius, so at most $\log \Delta$ many. We then compute among those candidates the closest to the center of b. Each query takes $\mathcal{O}(\log n + 1/\epsilon^{d-1})$ time. □

Next, we investigate how to predict a collision event induced by processing an update event. We observe

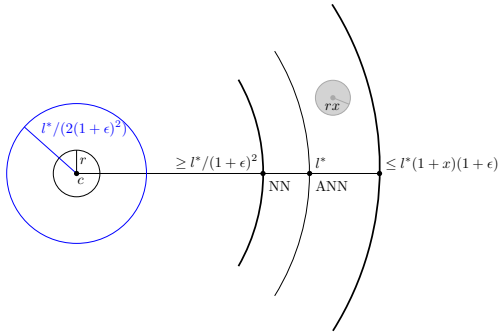


Figure 1: Illustration for the proof of Lemma 2. The annulus for given x is indicated by the two thick arcs.

that with l^* being the current ANN distance for a ball b, the current time is at least $l^*/(2(1+\epsilon)^2 r)$, as l^* is within a factor of $(1+\epsilon)$ of the former ANN l_0^* and this ANN led to the update event $(b, l_0^*/(2(1+\epsilon)r))$.

Lemma 2 *For a ball $b = (c, r)$ at time $t \geq l^*/(2(1+\epsilon)^2 r)$, with l^* being the ANN distance, the next collision with a ball $b' = (c', r')$, $r' \leq r$, can be predicted in time $\mathcal{O}(\log \Delta(\log n + 1/\epsilon^{d-1} + \max(\epsilon\Delta^d, \Delta^{d-1})))$ for $\epsilon \in]0, 1]$.*

Proof. In case the ANN ball b_{ANN} is *not* the next collision partner for b, there either is a ball even closer to b than the ANN or there is a ball further away than the ANN but with a radius proportionally larger.

The NN distance l is $\geq l^*/(1+\epsilon) \geq l^*/(1+\epsilon)^2$. So in the annulus centered at c with $r_1 = l^*/(1+\epsilon)^2$ and $r_2 = l^*$, there can be alternative collision partners. For a ball $b' = (c', r')$ with $l' = |cc'| > l^*$ to be a possible collision partner, the collision time has to be $\leq l^*/r$, as this is an upper bound on $t_{col}(b, b_{ANN})$. So we get $l'/(r+r') \leq l^*/r \Rightarrow l' \leq l^*(1 + r'/r)$. Note that we always have $r'/r \geq 1/\Delta$. For each $x = 1/2^i$, $i = 0, \cdots, \log r$, we issue an ARR query $B_{l^*(1+x)}(c)$ to the DS of the class containing radii rx. The result returns centers up to a distance of $l^*(1+x)(1+\epsilon)$. The main question now is how many balls with a radius of $\Omega(rx)$ can fit inside the annulus $r_1 = l^*/(1+\epsilon)^2$, $r_2 = l^*(1+x)(1+\epsilon)$ at time $t = l^*/(2(1+\epsilon)^2 r)$ (see Figure 1 for an illustration). The annulus volume is $V_a = \mathcal{O}(l^{*d}((1+x)^d(1+\epsilon)^d - 1/(1+\epsilon)^{2d}))$. The volume of a ball of radius rx at time t is $V_{b'} = \Omega(l^{*d} x^d/(1+\epsilon)^{2d})$ (and at least a constant fraction has to be inside the annulus). Hence, $V_a/V_{b'} = \mathcal{O}(((1+x)^d(1+\epsilon)^{3d} - 1)/x^d)$. Because of the -1 in the numerator, after expansion of the product $(1+x)^d(1+\epsilon)^{3d}$ all remaining terms are either divisible by ϵ or x. All coefficients are in $\mathcal{O}(2^{4d}) = \mathcal{O}(1)$ for fixed d and can therefore be neglected. As $\epsilon, x \in]0, 1]$, the largest of the summands after cancellation with the denominator is $\mathcal{O}(\epsilon/x^d)$ in case $\epsilon \geq 1/\Delta$, and $\mathcal{O}(1/x^{d-1})$ otherwise, and we

have $\mathcal{O}(d^2) = O(1)$ summands. As $x \geq 1/\Delta$, we get $\mathcal{O}(\max(\epsilon\Delta^d, \Delta^{d-1}))$ as an upper bound on the result size of one approximate RR query. As we issue at most $\log\Delta$ such queries and each of them costs $\mathcal{O}(\log n + 1/\epsilon^{d-1} + \max(\epsilon\Delta^d, \Delta^{d-1}))$, the running time bound follows. □

We observe that the time to predict the next collision is minimized in case $1/\epsilon^{d-1} = \epsilon\Delta^d = \Delta^{d-1}$ which is realized by $\epsilon = 1/\Delta$. Therefore, the time to compute an update as well as a collision is in $\mathcal{O}(\log\Delta(\log n + \Delta^{d-1}))$.

2.2 Bounding the number of events

Let n_u be the number of update events, and n_c the number of collision events inserted in the PQ in the course of the algorithm. We know that there are exactly $n-1$ 'real' collision events, that is, collision events with both balls being still alive at the time of extraction from the PQ. And we know that we initially insert n update events in the PQ.

So the question is how many update events lead to new update events and how many collision events in the PQ are affected by a real collision, i.e. one of the collision partners dies before the event time.

Lemma 3 *The number of times an update event (b,t) leads to the direct insertion of a new update event (b,t') is in $\mathcal{O}(n)$.*

Proof. Creation of a new update event (b,t') is only triggered if $t' > (1+\epsilon)t$. We observe that this can only happen if the real NN of b was eliminated. Otherwise, if the NN is alive and we get two ANN candidates, their distance difference towards c can never vary more than a factor of $(1+\epsilon)$. As proven in [1], only a constant number of balls can share the same NN. Hence every elimination of a ball can trigger at most a constant number of new updates. □

Lemma 4 *For any ball $b \in B$ and at any time of the algorithm, the number of collision events in the PQ involving b is bounded by a constant.*

Proof. The number of collision events with b as anchor is ≤ 1. So the interesting case is $b = (c,r)$ being the smaller of the two collision partners. We define $b^+ = (c^+, r^+)$ with $r^+ \geq r$ to be the collision partner with the largest center-to-center-distance k. In the ball $B_k(c^+)$, there cannot be centers of other balls with a radius $\in [r, r^+]$, as they would be the collision partner for b^+ instead of b. But there might be balls with a radius $> r^+$ centered in $B_k(c^+)$, as we only looked for the next collision of b^+ with a smaller or equal sized ball. The question is how many balls with a radius $> r^+$ can be centered in $B_k(c^+) \cap B_k(c)$ – as they might be other possible collision partners for b.

Let t be the time at which the collision event $(b^+, b, t_{col}(b^+, b))$ was inserted in the PQ. We know that $t \geq l^*/(2(1+\epsilon)r^+)$ with l^* being the ANN distance for b^+. Furthermore, we know that the distance to a smaller collision partner cannot exceed $2l \leq 2l^*$. So we have $k \leq 2l^*$, and we know that every ball with a radius $\geq r^+$ has to have an induced radius of $\geq l^*/(2(1+\epsilon)) \geq k/(4(1+\epsilon)) \geq k/8$. Every ball with a center in $B_k(c^+) \cap B_k(c)$ has at least a constant fraction of its volume inside as well, and the volume of $B_k(c^+) \cap B_k(c)$ is proportional to k as is the volume of the inscribed balls. Hence only a constant number of such balls can be centered there.

So $B_k(c^+)$ cuts away a constant fraction of $B_r(c)$ and we can show that only a constant number of other collision partners can be associated with that volume. We then repeat the argument for the collision partner with next smaller distance to c which was not accounted for, and so on. After constantly many repetitions, the whole volume of $B_r(c)$ has been covered. So in total, only a constant number of larger balls can have b as their current collision partner. □

It follows that in total only a linear number of collision events can lead to additional update event insertions. Hence the overall number of update and collision events is $\mathcal{O}(n)$.

Theorem 5 *The total running time of the algorithm is $\mathcal{O}(n\log\Delta(\log n + \Delta^{d-1}))$.*

Proof. We need $\mathcal{O}(n\log n)$ to build suitable NN+RR DS for the $\log\Delta$ classes. We then insert a linear number of update and collision events into the PQ (Lemma 3 and 4) and extract and process them, taking time $\mathcal{O}(n\log n)$ in total. Each of the events can be computed in time $\mathcal{O}(\log\Delta(\log n + \Delta^{d-1}))$ (Lemma 1 and 2). And we spend time $\mathcal{O}(n\log n)$ on deleting (and reinserting) points in data structures. So the total running time is $\mathcal{O}(n\log\Delta(\log n + \Delta^{d-1}))$. □

3 Considering the number of radii classes C

We now assume that there are $C \leq n$ different radii in the input. For $C = 1$, it automatically follows $\Delta = 1$. Hence our Δ-dependent algorithm described above has an asymptotically optimal running time of $\mathcal{O}(n\log n)$ for arbitrary fixed d.

3.1 Larger balls, higher priority

We now consider a special case, where the priorities are positively correlated with the radii, that is, for two balls with $p_a > p_b$ we have $r_a \geq r_b$ which excludes smaller balls having larger priorities.

We will now present an algorithm for $d = 2$ which runs in time $\mathcal{O}(Cn\log^{\mathcal{O}(1)} n)$. Note that if there were exact NN and RR data structures for $d \geq 3$ with a

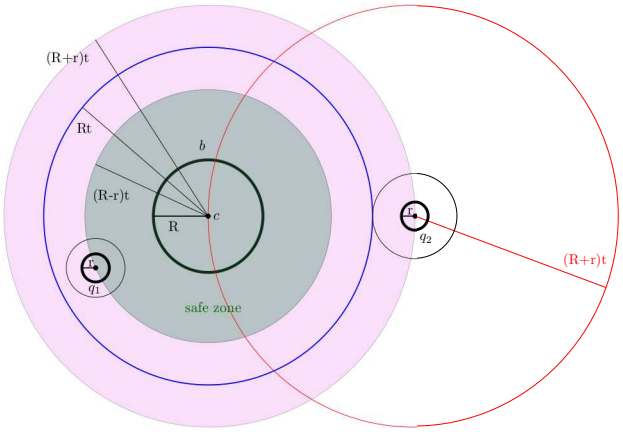


Figure 2: Illustration of the proof of Lemma 6.

running time of $\mathcal{O}(\log^{\mathcal{O}(1)} n + k)$, the same overall running time would apply.

The algorithm starts by partitioning the balls into their respective radii classes $R_1, R_2, \cdots, R_C$ with $R_i < R_{i+1}$. With n_i we refer to the number of balls in class R_i. Then, for the class R_C, we compute the elimination order within. As seen above, this takes time $\mathcal{O}(n_C \log n_C)$. Thereafter, the suffix of the elimination sequence is already known, as the elimination times for balls in class R_C cannot be influenced by balls with smaller radii. For all other classes R_i, there might be collisions with balls in some class $R_j, j \geq i$. We will detect these collision events efficiently in a top-down fashion.

Lemma 6 *For a class R_j with known elimination times, all possible collision events with balls in classes R_i for $i = 1, \cdots, j-1$ can be computed in time $\mathcal{O}((n_j C + n) \log^{\mathcal{O}(1)} n)$.*

Proof. We consider the balls in R_j in increasing order of their elimination sequence. Let b be the current ball with center c and elimination time t. For each class $R_i, i \leq j$ we assume that a RR-data structure is available.

Observe that all balls in R_i with their center being further than $(R_j + R_i)t$ away from c can never collide with b before its elimination. So we issue the RR $B_{(R_j+R_i)t}(c)$ to the data structure for class R_i to get all potential collision partners. This takes time $\mathcal{O}(\log^{\mathcal{O}(1)} n + k \log n)$ with $k = |B_{(R_j+R_i)t}(c)|$.

Now note that all balls in R_i with their center within $(R_j - R_i)t$ of c can only collide with b among all balls in R_j still to be considered, as no such ball can reach them before b does, see Figure 2 for an illustration (with $R_j = R$ and $R_i = r$). Therefore, we can simply compare the collision time with b with the current collision time assigned to those balls in R_i and update it if necessary.

For balls $q \in R_i$ with their center between $(R_j - R_i)t$ and $(R_j + R_i)t$, b or some other still alive ball $b' \in R_j$ is the collision partner in R_j. If b' collides earlier with q than b, the distance of the centers of b' and q has to be smaller than $(R_j + R_i)t$. But every such b' is alive at t and hence has a radius of $R_j t$ at the elimination time of b. Accordingly, a RR $B_{(R_j+R_i)t}(c(q))$ in the DS for R_j (containing only alive balls at time t), will return only a constant number of candidates to check for the real next collision (see again Figure 2). The DS for R_j is then maintained by deleting b, all elements in $B_{(R_j+R_i)t}(c(q))$ are removed from the datastructure for R_i. The overall running time for collision checks with balls in R_j is then composed of (1) n_j range queries per class $R_i, i < j$, taking time $\mathcal{O}(\log^{\mathcal{O}(1)} n + k \log n)$ each, with all k summing up to at most n, hence in total $\mathcal{O}(n_j C \log^{\mathcal{O}(1)} n + n \log n)$. (2) At most n range queries into the DS for R_j to identify possible collision partners for balls not in the safe zone, demanding time $\mathcal{O}(\log^{\mathcal{O}(1)} n)$ each as $k \in \mathcal{O}(1)$ here, and so in total taking $\mathcal{O}(n \log^{\mathcal{O}(1)} n)$ time. (3) Updating the RR-DS for R_j, consisting of n_j deletions, hence taking time $\mathcal{O}(n_j \log^{\mathcal{O}(1)} n)$ in total, similarly updating the RR-DS for all the R_j, consisting of $O(n)$ deletions, hence $\mathcal{O}(n \log^{\mathcal{O}(1)} n)$ in total. Summing up over all three components results in $\mathcal{O}((n_j C + n) \log^{\mathcal{O}(1)} n)$. □

By invoking Lemma 6 for class $R_{c-1}, R_{c-2}, \cdots$ one after the other, we achieve the following runtime:

Theorem 7 *Computing the elimination sequence for a ball tournament with C radii classes and priorities being proportional to the radii costs $\mathcal{O}(nC \log^{\mathcal{O}(1)} n)$.*

4 Conclusions and Future Work

Our results should be seen as a further step towards a subquadratic or ideally near-linear time algorithm for the ball tournament problem. Future work will focus on reducing or even getting rid of the dependency on C and Δ. But similar to the motorcycle graph problem, where no near-linear algorithm is known without further restrictions, this might be difficult to achieve.

References

[1] D. Bahrdt, M. Becher, S. Funke, F. Krumpe, A. Nusser, M. Seybold, and S. Storandt. Growing balls in R^d. In *Algorithm Engineering and Experiments (ALENEX) (to appear)*, 2017.

[2] S. Funke, F. Krumpe, and S. Storandt. Crushing disks efficiently. In *Proc. 27th Int. Workshop on Combinatorial Algorithms (IWOCA)*, pages 43–54, 2016.

[3] D. Mount and E. Park. A Dynamic Data Structure for Approximate Range Searching. In *Proc. 26th Ann. Symp. on Computational Geometry (SoCG)*, 2010.

[4] A. Vigneron and L. Yan. A faster algorithm for computing motorcycle graphs. *Discrete & Computational Geometry*, 52(3):492–514, 2014.

Triangles in Arrangements of Pseudocircles*

Stefan Felsner† Manfred Scheucher‡§

Abstract

Grünbaum conjectured that the number of triangular cells p_3 in digon-free arrangements of n pairwise intersecting pseudocircles is at least $2n-4$. We present examples to disprove this conjecture. With a recursive construction based on an example with 12 pseudocircles and 16 triangles we obtain a family with $p_3(\mathscr{A})/n \to 16/11 = 1.\overline{45}$. We conjecture that the lower bound $p_3 \geq 4n/3$ of Hershberger and Snoeyink is tight for infinitely many arrangements. For intersecting arrangements with digons we have $p_3 \geq 2n/3$, and conjecture that $p_3 \geq n-1$.

First counterexamples to Grünbaum's conjecture were found on the basis of an exhaustive enumeration of all arrangements of n intersecting pseudocircles for $n \leq 7$. It turned out that there is a unique digon-free intersecting arrangement $\mathscr{N}_6$ with $n = 6$ and only 8 triangles. This arrangement is a subarrangement of all minimizing examples for $n = 7, 8, 9$. We show that $\mathscr{N}_6$ is not circularizable, i.e., there is no equivalent arrangement of circles. These results suggest that Grünbaum's conjecture might be true for digon-free intersecting arrangements of circles.

1 Introduction

We study *simple*, *intersecting* arrangements of pseudocircles on the sphere. Here intersecting means that any two pseudocircles cross twice, while simple means that no three pseudocircles intersect in a common point.

An arrangement of pseudocircles is called *completely intersecting* if there are two cells, which are separated by each of the pseudocircles. Note that for every completely intersecting arrangement of pseudocircles there is a stereographic projection from the sphere to the plane such that those two cells are mapped to the outer cell and a cell, which lies "inside" every pseudocircle, respectively.

In an arrangement $\mathscr{A}$ of pseudocircles, we denote a cell with k crossings on its boundary as *k-cell* and let $p_k(\mathscr{A})$ be the number of k-cells of $\mathscr{A}$. As usual we call 2-cells *digons*, 3-cells *triangles*, 4-cells *quadrangles*, and 5-cells *pentagons*.

In his monograph [3] from 1972, Grünbaum states Conjecture 3.7: *Every (not necessarily simple) digon-free arrangement of n pairwise intersecting pseudocircles has at least $2n-4$ triangles.* Grünbaum also provides examples of arrangements with $n \geq 6$ pseudocircles and $2n-4$ triangles. Snoeyink and Hershberger [7] showed that every connected digon-free arrangement of n pseudocircles has at least $4n/3$ triangles. Felsner and Kriegel [2] observed that the bound from [7] also applies to non-simple intersecting digon-free arrangements and gave examples of arrangements showing that the bound is tight on this class.

In Section 2, we give counterexamples to Grünbaum's conjecture. A specific arrangement $\mathscr{N}_6$ of 6 pseudocircles appears as subarrangement in most of the known counterexamples. In Section 3, we show that $\mathscr{N}_6$ is not circularizable, i.e., representable by circles. This motivates the question, whether indeed Grünbaum's conjecture is true when restricted to intersecting arrangements of circles. In the course of the presentation, we offer some additional conjectures, e.g., in Subsection 2.1 where we discuss arrangements with digons.

In this paper (unless explicitly stated otherwise) the term *arrangement* is used as equivalent to *simple arrangement of pairwise intersecting pseudocircles*.

2 Arrangements with few Triangles

In this section, we discuss arrangements with few triangles. The main result is the following theorem, which disproves Grünbaum's conjecture.

Theorem 1 *The minimum number of triangles in digon-free arrangements of n pseudocircles is*

(i) 8 *for* $3 \leq n \leq 6$.

(ii) $\lceil \frac{4}{3}n \rceil$ *for* $6 \leq n \leq 14$.

(iii) $< \frac{16}{11}n$ *for all* $n = 11k+1$ *with* $k \in \mathbb{N}$.

The basis for this theorem was laid by exhaustive computations, which generated all simple arrangements of up to $n = 7$ pseudocircles. We generated all possible dual graphs of such arrangements, that is, the graph on the faces, where two vertices share an edge if the correspond faces share a common segment of a pseudocircle. Since counting arrangements

*Partially supported by DFG Grant FE 340/11-1 and ERC Advanced Research Grant no 267165 (DISCONV).

†Institut für Mathematik, Technische Universität Berlin, Germany, felsner@math.tu-berlin.de

‡Institute of Software Technology, Graz University of Technology, mscheuch@ist.tugraz.at

§Alfréd Rényi Institute of Mathematics, Hungarian Academy of Sciences, Budapest, Hungary

is also interesting, we digress to present the enumerative results in Table 1. The four rows of the table show the number of simple pseudocircle arrangements without fixed outer cell (sphere) and with fixed outer cell (plane). In both cases, we first state the numbers when digons are allowed and then the numbers of digon-free arrangements. The arrangements and more information can be found on our website [6].

n	2	3	4	5	6	7
sphere	1	2	8	278	145 058	447 905 202
+digon-free	0	1	2	14	2 131	3 012 972
plane	1	4	45	5 108	4 598 809	?
+digon-free	0	1	5	157	63 808	132 355 602

Table 1: Number of combinatorially different arrangements of n pseudocircles.

Starting with $n = 7$, we iteratively used arrangements with n pseudocircles and a small number of triangles and digons to generate arrangements with $n+1$ pseudocircles and the same property. Using this strategy, we found the arrangements from (i) and (ii) of Theorem 1. Details can be found at [6]. From [7] we know the lower bound: Every digon-free arrangement has at least $4n/3$ triangles.

A result of the computations was that the triangle-minimizing example for $n = 6$ is unique, i.e., there is a unique simple arrangement $\mathcal{N}_6$ with 6 pseudocircles and only 8 triangles. This arrangement is a subarrangement of each of the minimizing examples for $7 \le n \le 9$. The claim, that indeed we found all minimizing examples in this range, is justified by Lemma 2, which allows to quantify the range of pairs (p_2, p_3) of arrangements of n pseudocircles whose extension may yield a minimizing example for $n+1$. In particular, to get all arrangements with $n = 9$ and 12 triangles we only had to extend arrangements with $n = 7$ and $n = 8$, where $p_3 + 2p_2 \le 12$.

Lemma 2 *For any arrangement $\mathscr{A}$ and $C \in \mathscr{A}$, we have* $p_3(\mathscr{A}) + 2p_2(\mathscr{A}) \ge p_3(\mathscr{A} - C) + 2p_2(\mathscr{A} - C)$.

Proof. Consider a triangle of $\mathscr{A} - C$. After adding C, either the triangle remains untouched, or the triangle is split into a triangle and a quadrangle, or a digon is created in the region covered by the triangle. Now consider a digon of $\mathscr{A} - C$. After adding C, either there is a digon in this region or the digon has been split into two triangles. □

It turns out that $\mathcal{N}_6$ is a subarrangement of many arrangements, that violate Grünbaum's conjecture. In Section 3, we show that $\mathcal{N}_6$ is not circularizable, i.e., there is no equivalent arrangement of circles. This property is inherited by all arrangements, that have $\mathcal{N}_6$ as a subarrangement. For the examples with less than $2n - 4$ triangles, that do not contain a subarrangement equivalent to $\mathcal{N}_6$, we could not find realizations by circles. Therefore, the following weakening of Grünbaum's conjecture may be true.

Conjecture 1 (Weak Grünbaum Conjecture) *Every digon-free arrangement of n circles has at least $2n - 4$ triangles.*

We now come to the proof of (iii) of Theorem 1. The basis of the construction is an arrangement $\mathscr{A}_{12}$ with 12 pseudocircles and 16 triangles shown in Figure 1. This arrangement will be used iteratively in a 'merge' operation as described by the following lemma.

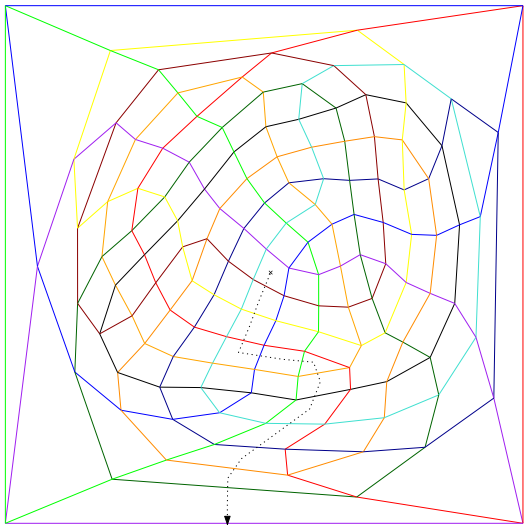


Figure 1: A digon-free, completely intersecting arrangement $\mathscr{A}_{12}$ of $n = 12$ pseudocircles with exactly 16 triangles. The dotted curve intersects every pseudocircle exactly once.

Lemma 3 *Let $\mathscr{A}$ and $\mathscr{B}$ be arrangements of $n_{\mathscr{A}}$ and $n_{\mathscr{B}}$ pseudocircles, respectively, and let $P_{\mathscr{A}}$ be a path in $\mathscr{A}$, that intersects every pseudocircle exactly once. If $P_{\mathscr{A}}$ traverses τ triangles of $\mathscr{A}$ and forms δ triangles with pairs of pseudocircles from $\mathscr{A}$, then there is an arrangement $\mathscr{C}$ of $n_{\mathscr{A}} + n_{\mathscr{B}} - 1$ pseudocircles with $p_3(\mathscr{C}) = p_3(\mathscr{A}) + p_3(\mathscr{B}) + \delta - \tau - 1$.*

Proof. Take a drawing of $\mathscr{A}$ and make a hole in the two cells, where the path $P_{\mathscr{A}}$ ends. This yields a drawing of $\mathscr{A}$ on a cylinder such that none of the pseudocircles is contractible. The path $P_{\mathscr{A}}$ connects the two boundaries of the cylinder. Now we stretch the drawing such that it becomes a narrow belt, where all intersections of pseudocircles take place in a small disk, which we call *belt-buckle*. This drawing of $\mathscr{A}$ is called a *belt drawing*. The construction is illustrated with the blue subarrangement in Figure 2.

Let B be a pseudocircle in $\mathscr{B}$ and let $\triangle$ be a triangle incident to B. Let b be the *edge* of B, which bounds $\triangle$. Specify a disk D, which is traversed by b and disjoint from all other edges of $\mathscr{B}$. Now replace B by a belt drawing of $\mathscr{A}$ in a small neighborhood of B such that the belt-buckle is drawn within D; see Figure 2.

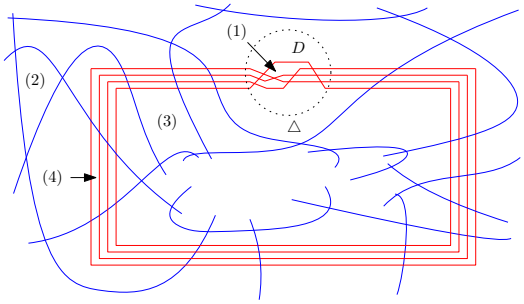


Figure 2: An illustration of the construction in Lemma 3. Pseudocircles of $\mathscr{A}$ ($\mathscr{B}$) are drawn red (blue).

The arrangement $\mathscr{C}$ obtained from the *merge* of $\mathscr{B}$ and $\mathscr{A}$ along B consists of $n_{\mathscr{A}} + n_{\mathscr{B}} - 1$ pseudocircles. Most of the cells of $\mathscr{C}$ are of one of the following four types: (1) All boundary edges belong to pseudocircles of $\mathscr{A}$. (2) All boundary edges belong to pseudocircles of $\mathscr{B}$. (3) All boundary edges but one belong to pseudocircles of $\mathscr{B}$ and the remaining edge belongs to the first or the last pseudocircle of $\mathscr{A}$ intersected by $P_{\mathscr{A}}$. These cells correspond to cells of $\mathscr{B}$ with a boundary edge on B. (4) Quadrangular cells, whose boundary edges alternatingly belong to $\mathscr{A}$ and $\mathscr{B}$.

From the cells of $\mathscr{B}$, only $\triangle$ and the other cell containing b (which is not a triangle since $\mathscr{B}$ is simple) have not been taken into account. In $\mathscr{C}$, the corresponding two cells have at least two boundary edges from $\mathscr{B}$ and at least two from $\mathscr{A}$. Consequently, neither of the two cells are triangles. The remaining cells of $\mathscr{C}$ have been created by inserting $P_{\mathscr{A}}$ into $\mathscr{A}$. To be precise, the role of $P_{\mathscr{A}}$ in these cells is taken by one of the two boundary pseudocircles of $\triangle$ other than B. There are δ triangles among these cells, but τ of these are obtained because $P_{\mathscr{A}}$ traverses a triangle of $\mathscr{A}$. All other triangles of $\mathscr{C}$ have a corresponding triangle in $\mathscr{A}$ or $\mathscr{B}$, except for $\triangle$, which does not occur in this correspondence. Altogether, there are $p_3(\mathscr{A}) + p_3(\mathscr{B}) + \delta - \tau - 1$ triangles in $\mathscr{C}$. □

Proof of Theorem 1(iii). We use $\mathscr{A}_{12}$, the arrangement shown in Figure 1, in the role of $\mathscr{A}$ for our recursive construction. The dotted path in the figure is used as $P_{\mathscr{A}}$ with $\delta = 2$ and $\tau = 1$. Starting with $\mathscr{C}_1 = \mathscr{A}_{12}$ and defining $\mathscr{C}_{k+1}$ as the merge of $\mathscr{C}_k$ and $\mathscr{A}_{12}$, we construct a sequence $\{\mathscr{C}_k\}_{k\in\mathbb{N}}$ of arrangements with $n(\mathscr{C}_k) = 11k + 1$ pseudocircles and $p_3(\mathscr{C}_k) = 16k$ triangles. The fraction $16k/(11k+1)$ is increasing as k increases with limit $16/11 = 1.\overline{45}$. □

We remark that using other arrangements from Theorem 1(ii) (which also admit a path with $\delta = 2$ and $\tau = 1$) in the recursion, we obtain arrangements with $p_3 \leq \lceil \frac{16}{11} n \rceil$ triangles for all $n \geq 6$.

Since the lower bound $\lceil \frac{4}{3} n \rceil$ is tight for $6 \leq n \leq 14$, we believe that the following is true:

Conjecture 2 *There are infinitely digon-free arrangements $\mathscr{A}$ with $p_3(\mathscr{A}) = \lceil 4n/3 \rceil$.*

2.1 Arrangements with Digons

Concerning arrangements with digons, we know of two constructions for families of arrangements with only $n - 1$ triangles. An example is shown in Figure 3.

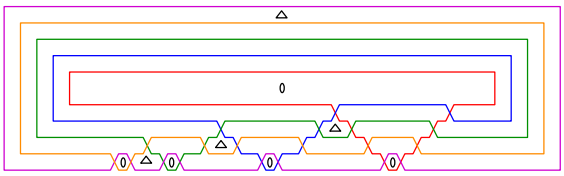


Figure 3: An illustration of an arrangement of $n = 5$ pseudocircles with n digons and $n - 1$ triangles.

Using ideas based on sweeps (cf. [7]), we can show that every pseudocircle is incident to at least two triangles. This implies the following theorem:

Theorem 4 *Every arrangement of $n \geq 3$ pseudocircles has at least $2n/3$ triangles.*

Since for $3 \leq n \leq 7$ every arrangement has at least $n - 1$ triangles, we believe that the following is true:

Conjecture 3 *Every arrangement of $n \geq 3$ pseudocircles has at least $n - 1$ triangles.*

3 Non-circularizable Arrangements

Little is known about *circularizability*, i.e., deciding whether a given arrangement of pseudocircles is isomorphic to an arrangement of circles. Edelsbrunner and Ramos [1] proved non-circularizability of an arrangement of 6 pseudocircles with digons. Linhart and Ortner [5] found a non-intersecting arrangement of 5 pseudocircles with digons, that is non-circularizable. Kang and Müller [4] proved that all arrangements with at most 4 pseudocircles are circularizable and that deciding circularizability is NP-hard in general.

Having generated all intersecting arrangements with $n \leq 7$, we used a randomized procedure to see, which of them are realizable as circle arrangement. After realizing some remaining hard instances with $n = 5$ by hand, we now have:

Proposition 5 *The arrangement $\mathscr{N}_5$ shown in Figure 4(a) is the unique non-circularizable arrangement among the 278 equivalence classes of intersecting arrangements of $n = 5$ pseudocircles.*

Proof (Sketch). Since we have realizations of all 278 intersecting arrangements of $n = 5$ pseudocircles except $\mathscr{N}_5$, it remains to show that $\mathscr{N}_5$ is not circularizable. Suppose for a contradiction that there is

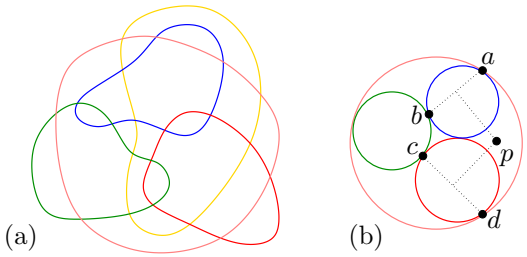


Figure 4: (a) The unique intersecting non-circularizable arrangement $\mathcal{N}_5$ of 5 pseudocircles. (b) An illustration of the proof of Proposition 5.

an equivalent arrangement $\mathcal{A}$ of circles. Shrink the red, green, and blue circle into their interior so that they touch each other and they all touch the pink circle; see Figure 4(b). Four of the touching points have been labeled. The bisectors of the chords ab and cd intersect in a point p, which is equidistant to a, b, c, and d. Hence, there is a circle C with center p, which is incident to each of the four points. Since the four labeled points are in four of the digons of $\mathcal{A}$, we know that the yellow circle of $\mathcal{A}$ has a and c in its interior but b and d in its exterior. Since on C, the counter-clockwise order of the four points is a, b, c, d, there is no circle with the properties needed for the yellow circle of $\mathcal{A}$. A contradiction. □

The proposition together with the work of Kang and Müller implies that all digon-free intersecting arrangements of at most 5 pseudocircles are circularizable. For $n = 6$ there are digon-free intersecting arrangements, which are non-circularizable. Figure 5 shows such an arrangement, which we denote as $\mathcal{N}_6$. The arrangement $\mathcal{N}_6$ is the unique arrangement for $n = 6$ minimizing the number of triangles, and, since $\mathcal{N}_6$ occurs as a subarrangement of every triangle-minimizing arrangement for $n = 7, 8, 9$, also neither of those arrangements is circularizable. From the 2131 digon-free intersecting arrangements of 6 pseudocircles 2128 are circularizable and 3 are not. In the following we sketch the proof of the non-circularizability of the arrangement $\mathcal{N}_6$. All realizations and the two additional non-circularizable arrangements can be found at [6].

Proposition 6 *The arrangement $\mathcal{N}_6$, as depicted in Figure 5, is non-circularizable.*

Proof (Sketch). Suppose for a contradiction that there is an equivalent arrangement of circles on the unit sphere. Choose a point in each of the eight triangles on the sphere and label them with letters as in Figure 5. Now embed $\mathbb{R}^3$ as an affine subspace into $\mathbb{R}^4$ such that a, b, c, d are mapped onto the standard basis vectors $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \mathbf{e}_4$ in this order. On the sphere, the circle through a, b, c separates d and z. This implies that the first three components z_1, z_2, z_3 of z are positive and that z_4 is negative. Similarly, the unique negative components of w, x, y are w_1, x_2, and y_3, respectively. Next, for each circle of the arrangement, we consider the determinant of the four points in the incident triangles. E.g., for the green circle, we look at $\det(abwx)$. Geometric considerations allow us to argue that $\det(abwx)$ is positive. Therefore, $w_3x_4 > w_4x_3$. In an analogous manner, we obtain:

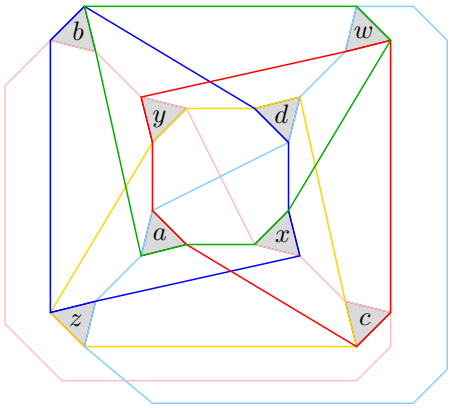


Figure 5: The unique digon-free intersecting arrangement $\mathcal{N}_6$ of 6 pseudocircles with 8 triangles. This arrangement is non-circularizable.

$$\begin{aligned}
\text{green}: &\quad \det(abwx) > 0\,; &\quad w_3x_4 &> w_4x_3\\
\text{red}: &\quad \det(cayw) > 0\,; &\quad w_4y_2 &> w_2y_4\\
\text{light blue}: &\quad \det(adwz) > 0\,; &\quad w_2z_3 &> w_3z_2\\
\text{pink}: &\quad \det(cbyx) > 0\,; &\quad x_1y_4 &> x_4y_1\\
\text{blue}: &\quad \det(bdxz) > 0\,; &\quad x_3z_1 &> x_1z_3\\
\text{yellow}: &\quad \det(dczy) > 0\,; &\quad y_1z_2 &> y_2z_1
\end{aligned}$$

The negative values do not show up in the inequalities. Moreover, if we take the product of the left-hand-sides and right-hand-sides, resp., we obtain the same value on both sides of the inequality – a contradiction. □

References

[1] H. Edelsbrunner and E. A. Ramos. Inclusion-exclusion complexes for pseudodisk collections. *Discrete & Computational Geometry*, 17:287–306, 1997.

[2] S. Felsner and K. Kriegel. Triangles in Euclidean arrangements. In *Proc. WG*, volume 1517 of *LNCS*, pages 137–148. Springer, 1998.

[3] B. Grünbaum. *Arrangements and Spreads*, volume 10 of *RCSM*. AMS, 1972 (reprinted 1980).

[4] R. J. Kang and T. Müller. Arrangements of pseudocircles and circles. *Discrete & Computational Geometry*, 51:896–925, 2014.

[5] J. Linhart and R. Ortner. An arrangement of pseudocircles not realizable with circles. *Beiträge zur Algebra und Geometrie*, 46:351–356, 2005.

[6] M. Scheucher. http://www.ist.tugraz.at/scheucher/arrangements_of_pseudocircles.

[7] J. Snoeynik and J. Hershberger. Sweeping arrangements of curves. In Goodman, Pollack, and Steiger, editors, *Discrete and computational geometry*, DIMACS Ser. DMTCS vol 6, pages 309–349. AMS, 1991.

Arrangements of Approaching Pseudo-Lines

Stefan Felsner* Alexander Pilz†

Abstract

We consider arrangements of n pseudo-lines in the Euclidean plane where each pseudo-line ℓ_i is represented by a bi-infinite connected x-monotone curve $f_i(x)$, $x \in \mathbb{R}$, s.t. for any two pseudo-lines ℓ_i and ℓ_j with $i < j$, the function $x \mapsto f_j(x) - f_i(x)$ is monotonically decreasing (i.e., the pseudo-lines approach each other until they cross, and then move away from each other). We show that such *arrangements of approaching pseudo-lines*, under some aspects, behave similar to arrangements of lines, while for other aspects, they share the freedom of general pseudo-line arrangements. For the former, we prove that there are arrangements of pseudo-lines that are not realizable with approaching pseudo-lines. For the latter, we show: (i) There are $2^{\Theta(n^2)}$ isomorphism classes of arrangements of approaching pseudo-lines (while there are only $2^{\Theta(n \log n)}$ isomorphism classes of line arrangements). (ii) It can be decided in polynomial time whether an allowable sequence is realizable by an arrangement of approaching pseudo-lines.
Furthermore, arrangements of approaching pseudo-lines can be transformed into each other by flipping triangular cells, i.e., they have a connected flip graph, and any bichromatic such arrangement contains a bichromatic triangular cell.

1 Introduction

Arrangements of lines and, in general, arrangements of hyperplanes are paramount data structures in computational geometry, whose combinatorial properties have been extensively studied, partially motivated by the point-hyperplane duality. Many combinatorial properties of line arrangements are shared with (and actually were developed through) their generalization to pseudo-line arrangements. While pseudo-lines can be considered either as combinatorial or geometric objects, they lack certain geometric properties that may come in handy for proofs, as in the following example, which motivated the research presented here.

Consider a finite set of lines that are either red or blue, no two of them parallel and no three of them passing through the same point. Every such arrangement has a bichromatic triangular cell, i.e., an empty triangle defined both by red and blue lines. This can be shown using a distance argument similar to Kelly's proof of the Sylvester-Gallai theorem (see, e.g., [2, p. 73]). We sketch another nice proof. Suppose w.l.o.g. that no two crossings in the arrangement have the same x-coordinate (otherwise, slightly rotate the plane), and that there is a blue crossing above a red line. Continuously translate the arrangement of red lines in positive y-direction while keeping the arrangement of blue lines in place until a crossing lies on a line. Note that the crossing is monochromatic and the line has a different color. Hence, just before that event, the crossing is in the vicinity of the line and thus the three lines involved form a bichromatic triangle. However, until then, the combinatorial structure of the arrangement has not changed and thus the arrangement initially contained such a triangle.

This and all the known proofs for existence of a bichromatic triangle do not generalize to pseudo-line arrangements. Actually the question for pseudo-line arrangements is by now open for several years. The crucial property of lines used in the above argument is that shifting a subset of the lines vertically again yields an arrangement, i.e., the the shift does not introduce multiple crossings. We were wondering whether any pseudo-line arrangement can be drawn s.t. this property holds. In this abstract, we show that this is not true and that arrangements where this is possible constitute an interesting class of pseudo-line arrangements.

We define an *arrangement of pseudo-lines* as a finite family of x-monotone bi-infinite connected curves (called *pseudo-lines*) in the Euclidean plane s.t. each pair of pseudo-lines intersects in exactly one point, at which they cross. For simplicity, we consider the n pseudo-lines $\{\ell_1, \ldots, \ell_n\}$ to be indexed from 1 to n in top-bottom order at left infinity.[1] A pseudo-line arrangement is *simple* if no three pseudo-lines meet in one point; if in addition no two pairs of pseudo-lines cross at the same x-coordinate we call it *x-simple.*

An *arrangement of approaching pseudo-lines* is an arrangement of pseudo-lines where each pseudo-line ℓ_i

*Institut für Mathematik, Technische Universität Berlin, felsner@math.tu-berlin.de. Partially supported by DFG grant FE 340/11–1.

†Department of Computer Science, ETH Zürich, alexander.pilz@inf.ethz.ch. Supported by a Schrödinger fellowship, Austrian Science Fund (FWF): J-3847-N35.

[1] Pseudo-line arrangements are often studied in the real projective plane, with pseudo-lines being simple closed curves that do not separate the projective plane. All arrangements are isomorphic to x-monotone arrangements [10]. As x-monotonicity is crucial for our setting and the line at infinity plays a special role, we use the above definition.

is represented by function-graph $f_i(x)$, defined for all $x \in \mathbb{R}$, s.t. for any two pseudo-lines ℓ_i and ℓ_j with $i < j$, the function $x \mapsto f_j(x) - f_i(x)$ is monotonically decreasing (i.e., the pseudo-lines approach each other until they cross, and then they move away from each other). For most of our results, we may, as shown in the full version, consider the pseudo-lines to be *strictly approaching*, i.e., the function is strictly decreasing. For simplicity, we may sloppily call arrangements of approaching pseudo-lines *approaching arrangements*.

For two pseudo-lines, having a decreasing signed distance by increasing x-value is a property that is not maintained by a projective transformation, even if it maintains the vertical direction. We thus emphasize that approaching arrangements are defined in the Euclidean plane. This contrasts with related work that often considers pseudo-line arrangements in the projective plane. The special features of approaching arrangements are taken into account with the following notions of equivalence. Two pseudo-line arrangements are *sweep-equivalent* iff a sweep with a vertical line meets the crossings in the same order. Two pseudo-line arrangements are *vertically isomorphic* if there is an isomorphism of their face lattices (i.e., they dissect the Euclidean plane in a combinatorially equivalent way) that also respects the indexing of the pseudo-lines, i.e., their vertical order at left infinity. The reader may already have noticed that sweep-equivalence captures the allowable sequence of pseudo-line arrangements. For these also, the vertical direction plays a special role. An *allowable sequence* is a sequence of permutations in which (i) a permutation is obtained from the previous one by the reversal of one or more non-overlapping substrings, and (ii) each pair is reversed exactly once.[2] An allowable sequence is *simple* if two adjacent permutations differ by the reversal of exactly two adjacent elements. Hence, the permutations in which a vertical sweep line intersects the pseudo-lines of an arrangement gives an allowable sequence, starting with the identity permutation $I = \{1, \ldots, n\}$ of the pseudo-line's indices. The arrangement is said to *realize* that allowable sequence.

In this abstract, we identify various notable properties of approaching arrangements. In Section 2, we show how to modify approaching arrangements and how to decide whether an arrangement is sweep-equivalent to an approaching arrangement in polynomial time. In the following section, we provide arrangements without sweep-equivalent and vertically isomorphic approaching arrangements in the next section. Further, we show that there are asymptotically as many different approaching arrangements as pseudo-line arrangements.

Related work. Restricted representations of pseudo-line arrangements have been considered already at the early beginning of this concept. Goodman [8] considers their representation as *wiring diagrams*, and there are results on drawing arrangements as convex polygonal chains with few bends [6] and on small grids [5]. Any pseudo-line arrangement can be represented in these ways. Goodman and Pollack [11] consider the arrangements whose pseudo-lines are the function-graphs of polynomial functions with bounded degree. In particular, they give bounds on the degree necessary to represent all isomorphism classes of pseudo-line arrangements. Generalizing our setting to higher dimensions (by requiring that any pseudo-hyperplane can be translated vertically while maintaining that the family of hyperplanes is an arrangement) tells us that such approaching arrangements are representations of *Euclidean oriented matroids*, which occur in pivot rules for oriented matroid programming (see [4, Chapter 10]).

2 Manipulating approaching arrangements

One essential tool we use is the transformation of arrangements of general approaching pseudo-lines to pseudo-lines that are piecewise linear, similar to the transformation of pseudo-line arrangements to sweep-equivalent wiring diagrams.

Lemma 1 *For any arrangement of approaching pseudo-lines, there is a sweep-equivalent arrangement of approaching polygonal curves (starting and ending with a ray). If the allowable sequence of the arrangement is simple, then there exists such an arrangement without crossings at the bends of the polygonal curves.*

Proof. (Sketch.) We can place a vertical 'helper-line' at every crossing of the arrangement. Connect the intersection points of each pseudo-line with adjacent helper-lines by segments. (If the initial curves were approaching, these segments are as well.) This results in a sweep-equivalent arrangement of approaching polygonal curves. To complete the construction, we appropriately add rays in negative and positive x-direction. □

Actually we could have extended the segments from the first and the last vertical slab, respectively, as these segments are approaching and thus the crossings of the supporting lines of the segments in the, say, first vertical interval are not to the left of the first vertical

[2] In the seminal work by Goodman and Pollack [9], "allowable sequences" have been defined as *periodic* sequences of permutations, where (i) a permutation is obtained from the previous one by the reversal of one or more non-overlapping substrings, and (ii) after the reversal of a pair ij, all other pairs are reversed before reversing i and j again [9]. This sequence is fully defined by a half-period, and we follow the frequent approach of calling that half-period an allowable sequence, as, e.g., in [4, p. 264]. Goodman and Pollack call arrangements with the same allowable sequence *combinatorially equivalent* [10].

helper-line. For the segments in the slabs, only the relative order of their slopes is relevant to maintain the "approaching" property.

Consider again the construction in the previous proof. After fixing the intersection points with the helper-lines, we may change the x-coordinate of these; as long as we maintain their relative order, the arrangement remains sweep-equivalent. Further, we may shift all the points on a helper-line up or down without alternating the combinatorial structure. We use this freedom for our next result, where we show that the intersection points with the helper lines can be obtained by a linear program. Asinowski [3] defines a *suballowable sequence* as a sequence obtained from an allowable sequence by removing an arbitrary number of permutations from it. An arrangement thus realizes a suballowable sequence if we can obtain this suballowable sequence from its allowable sequence.

Theorem 2 *Given a suballowable sequence, we can decide in polynomial time whether there is an arrangement of approaching pseudo-lines with such a sequence.*

Let us emphasize that deciding whether an allowable sequence is realizable by a line arrangement is an $\exists\mathbb{R}$-hard problem [13], and thus not even known to be in NP. While we do not have a polynomial-time algorithm for deciding whether there is a vertically isomorphic approaching arrangement for a given pseudo-line arrangement, Theorem 2 tells us that the problem is in NP, as we can give the order of the crossings encountered by a sweep as a certificate for a realization. The corresponding problem for lines is also $\exists\mathbb{R}$-hard [14].

The following observation is the main property that makes approaching pseudo-lines interesting.

Observation 1 *Given an arrangement of strictly approaching pseudo-lines, any vertical translation of any pseudo-line results again in an arrangement of strictly approaching pseudo-lines.*

Lemma 3 *For any simple approaching arrangement that is not x-simple, there exists an approaching arrangement that is sweep-equivalent apart from the crossings sharing the x-coordinate.*

For pseudo-line arrangements, Ringel's homotopy theorem [15] tells us that one can be transformed to any other by homeomorphisms of the plane and so-called *triangle flips*, where a pseudo-line is moved over the crossing between two other ones. For the subset of arrangements of approaching pseudo-lines, the result can be specialized in a way that the intermediate arrangements are also approaching. We first show the following specialization of Ringel's isotopy result [15].

Lemma 4 *Two sweep-equivalent arrangements of approaching pseudo-lines can be transformed into each other by a homeomorphism of the plane s.t. all intermediate arrangements are sweep-equivalent and consist of approaching pseudo-lines.*

Theorem 5 *Given two simple arrangements of approaching pseudo-lines, one can be transformed to the other by homeomorphisms of the plane and triangle swaps s.t. all intermediate arrangement are approaching.*

We note that the relative position of the crossings may change during this process. Also, our proof requires the arrangement to be simple.

Vertically translating pseudo-lines now allows us to prove a restriction of our motivating question.

Theorem 6 *Any simple arrangement of approaching red and blue pseudo-lines contains a triangular cell that is bounded by both a red and a blue pseudo-line.*

3 Properties

Considering the freedom one has in constructing approaching arrangements, one may wonder whether actually all pseudo-line arrangements are sweep-equivalent to approaching arrangements. However, this is not true, as we will see in this section. We use the following lemma, that can easily be shown using the construction for Lemma 1.

Lemma 7 *Given a simple suballowable sequence of permutations (I, π_1, π_2), where I is the identity permutation, the suballowable sequence is realizable with an arrangement of approaching pseudo-lines if and only if it is realizable as a line arrangement.*

Asinowski [3] provided such a suballowable sequence on six lines that is not realizable as a line arrangement.

Corollary 8 *There exist simple suballowable sequences that are not realizable as arrangements of approaching pseudo-lines.*

In Figure 1, we modify the construction to an arrangement not having an isomorphic approaching arrangement. The resulting object is a simple pseudo-line arrangement, and each vertically isomorphic arrangement contains Asinowski's sequence.

Corollary 9 *There are pseudo-line arrangements for which there exists no vertically isomorphic arrangement of approaching pseudo-lines.*

Aichholzer et al. [1] construct a suballowable sequence (I, π_1, π_2) on n lines s.t. all line arrangements realizing them require slope values that are exponential in the number of lines. Thus, also vertex coordinates in a polygonal representation as an approaching arrangement are exponential in n (but their size is not).

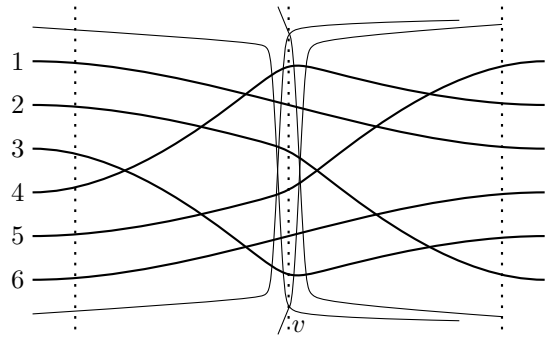


Figure 1: A part of a six-element pseudo-line arrangement (bold) whose suballowable sequence (indicated by the vertical lines) is non-realizable (adapted from [3, Fig. 4]). Adding the four thin pseudo-lines crossing in the vicinity of vertical v enforces that the allowable sequence of any vertically isomorphic arrangement contains the six-element permutation indicated by v.

Even though we have non-realizability results for approaching arrangements, their number is much larger than the number of arrangements of lines.

Theorem 10 *There exist $2^{\Theta(n^2)}$ isomorphism classes of simple arrangements of n approaching pseudo-lines.*

As there are only $2^{\Theta(n \log n)}$ isomorphism classes of simple line arrangements [12], we see that we have way more arrangements of approaching pseudo-lines. While the number of allowable sequences is $2^{\Theta(n^2 \log n)}$ [16], there are only at most n^{8n} combinatorially different line arrangements [12]. So arrangements of approaching pseudo-lines also differ in this setting, as the previous proof shows. However, we do not know whether our bound obtained via isomorphism classes is already asymptotically tight for allowable sequences.

Concerning further aspects where approaching arrangements are similar to line arrangements, we make the following conjecture, whose analogues hold for lines but not for pseudo-lines [7]. But so far we were unable to give a full proof.

Conjecture 1 *Every arrangement of n approaching pseudo-lines has at least $n-2$ triangular cells.*

4 Conclusion

In this paper, we introduced a type of pseudo-line arrangements that generalize line arrangements, but still retain certain geometric properties. One of the main algorithmic open problems is deciding the realizability of a pseudo-line arrangement as a vertically isomorphic approaching arrangement. Further, we do not know how projective transformations influence this realizability. The concept can be generalized to higher dimensions. Apart from the properties we already mentioned in the introduction, we are not aware of further non-trivial observations. Eventually, we hope for this concept to shed more light on the differences between pseudo-line arrangements and line arrangements.

References

[1] O. Aichholzer, T. Hackl, S. Lutteropp, T. Mchedlidze, A. Pilz, and B. Vogtenhuber. Monotone simultaneous embeddings of upward planar digraphs. *J. Graph Algorithms Appl.*, 19(1):87–110, 2015.

[2] M. Aigner and G. M. Ziegler. *Proofs from THE BOOK*. Springer, 5th edition, 2014.

[3] A. Asinowski. Suballowable sequences and geometric permutations. *Discrete Math.*, 308(20):4745–4762, 2008.

[4] A. Björner, M. Las Vergnas, B. Sturmfels, N. White, and G. Ziegler. *Oriented matroids*, volume 46 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, 1993.

[5] D. Eppstein. Drawing arrangement graphs in small grids, or how to play planarity. *J. Graph Algorithms Appl.*, 18(2):211–231, 2014.

[6] D. Eppstein, M. van Garderen, B. Speckmann, and T. Ueckerdt. Convex-arc drawings of pseudolines. *CoRR*, abs/1601.06865, 2016.

[7] S. Felsner and K. Kriegel. Triangles in Euclidean arrangements. *Discrete Comput. Geom.*, 22(3):429–438, 1999.

[8] J. E. Goodman. Proof of a conjecture of Burr, Grünbaum, and Sloane. *Discrete Math.*, 32(1):27–35, 1980.

[9] J. E. Goodman and R. Pollack. A theorem of ordered duality. *Geom. Dedicata*, 12:63–74, 1982.

[10] J. E. Goodman and R. Pollack. Semispaces of configurations, cell complexes of arrangements. *J. Combin. Theory Ser. A*, 37(3):257–293, 1984.

[11] J. E. Goodman and R. Pollack. Polynomial realization of pseudoline arrangements. *Commun. Pure Appl. Math.*, 38(6):725–732, 1985.

[12] J. E. Goodman and R. Pollack. Upper bounds for configurations and polytopes in R^d. *Discrete Comput. Geom.*, 1:219–227, 1986.

[13] U. Hoffmann. *Intersection graphs and geometric objects in the plane*. PhD thesis, Technische Universität Berlin, 2016.

[14] N. E. Mnëv. The universality theorems on the classification problem of configuration varieties and convex polytope varieties. In O. Y. Viro, editor, *Topology and Geometry—Rohlin Seminar*, volume 1346 of *Lecture Notes Math.*, pages 527–544. Springer, 1988.

[15] G. Ringel. Teilungen der Ebene durch Geraden oder topologische Geraden. *Math. Z.*, 64:79–102, 1956.

[16] R. P. Stanley. On the number of reduced decompositions of elements of Coxeter groups. *European J. Combin.*, 5:359–372, 1984.

Dushnik-Miller dimension of TD-Delaunay complexes*

Daniel Gonçalves[a] and Lucas Isenmann[a]

[a]*LIRMM, CNRS & Université de Montpellier, 161 rue Ada, 34095 Montpellier Cedex 5, France,*

Abstract

TD-Delaunay graphs, where TD stands for triangle-distance, were introduced by Chew and Drysdale [3]. These graphs are obtained from a variation of Delaunay triangulation consisting in replacing circles by homothetic triangles. In [4] the authors noticed that every triangulation is the TD-Delaunay graph of a set of points in $\mathbb{R}^2$, and conversely. It seems natural to study the generalization of this property in higher dimensions. Such a generalization is obtained by replacing equilateral triangles by regular simplexes in dimension d. The abstract simplicial complexes obtained from a TD-Delaunay complex in dimension d are of Dushnik-Miller dimension at most d. The converse holds for $d = 2$ and 3 and it was conjectured to hold for larger d [5] (See also [6]). Here we disprove the conjecture already for $d = 4$.

1 Dushnik-Miller dimension of simplicial complexes

An abstract simplicial complex Δ with vertex set V is a set of subsets of V which is closed by inclusion (*i.e.* $\forall Y \in \Delta$, $X \subseteq Y \Rightarrow X \in \Delta$). A face of Δ is an element of Δ. A facet of Δ is a maximal element of Δ according to the inclusion order.

Definition 1 *A d-representation R of a set V is given by d linear orders $<_1, \dots, <_d$ on V. Let R be a d-representation. We define $\Sigma(R)$ as the set of subsets X of V such that*

$$\forall v \in V, \exists i \in \{1, \dots, d\} : \forall x \in X, x \leq_i v$$

It is easy to show that $\Sigma(R)$ is an abstract simplicial complex. An example is the following 3-representation of $\{1, 2, 3, 4, 5\}$:

$<_1$	1	2	5	4	3
$<_2$	3	2	1	4	5
$<_3$	5	4	3	2	1

The corresponding complex $\Sigma(R)$ is given by the facets $\{1, 2\}, \{2, 3, 4\}$ and $\{2, 4, 5\}$. For example $\{1, 2, 3\}$ is not in $\Sigma(R)$ as 2 does not dominate $\{1, 2, 3\}$ in any order.

The Dushnik-Miller dimension of an abstract simplicial complex Δ is the minimum number d such that Δ is included in $\Sigma(R)$ for a d-representation R of V [1].

2 TD-Delaunay complexes

For any integer d, let H_d be the $(d-1)$-dimensional hyperplane of $\mathbb{R}^d$ defined by $\{x = (x^1, \dots, x^d) \in \mathbb{R}^d : x^1 + \dots + x^d = 1\}$. Let a *regular simplex* of H_d be any set $\{u \in H_d : u^1 \leq c^1, \dots, u^d \leq c^d\}$ for a $c \in \mathbb{R}^d$ such that $\sum_{i=1}^{d} c^i \geq 1$. In this context, a point set $\mathcal{P} \subset H_d$ is in *general position* if for any two vertices $x, y \in \mathcal{P}$, $x^i \neq y^i$ for every $1 \leq i \leq d$.

Definition 2 ([3]) *Given a set $\mathcal{P}$ of points of H_d in general position, let the TD-Delaunay complex of $\mathcal{P}$, denoted $\tau(\mathcal{P})$, be the simplicial complex with vertex set $\mathcal{P}$ defined as follows. A subset $F \subseteq \mathcal{P}$ is a face of $\tau(\mathcal{P})$ if and only if there exists a regular simplex S such that $S \cap \mathcal{P} = F$ and such that no point of $\mathcal{P}$ is in the interior of S.*

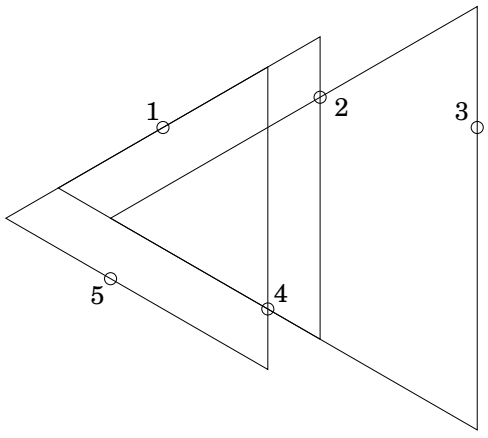


Figure 1: An example of a TD-Delaunay complex given by the facets $\{1, 4, 5\}, \{1, 2, 4\}$ and $\{2, 3, 4\}$

Definition 3 (TD-Delaunay system) *Given a d-representation R of a set V, let the* TD-Delaunay system *of R be the following linear system of inequalities.*

*This research is partially supported by the ANR GATO, under contract ANR 16 CE40 0009.

For every vertex u we define variables $u^1, \dots, u^d$. For every edge uv of $\Sigma(R)$ we have the constraint

$$\begin{cases} u^i - v^i < 0 \text{ if } u <_i v \\ v^i - u^i < 0 \text{ otherwise} \end{cases}$$

Then we replace each occurrence of u^d by $1 - u^1 - \dots - u^{d-1}$ for every $u \in V$. We note A_R the matrix corresponding to this system, that is the matrix such that $A_R x < 0$ for some vector x if and only if the TD-Delaunay system has a solution.

Example 4 *We consider the following 3-representation of $\{a, b, c\}$:*

$<_1$	b	c	a
$<_2$	a	c	b
$<_3$	a	b	c

The complex $\Sigma(R)$ is given by the facets $\{a, b, c\}$. The TD-Delaunay system of R is the following system:

$$\begin{cases} b^1 - a^1 < 0 \\ a^2 - b^2 < 0 \\ (1 - a^1 - a^2) - (1 - b^1 - b^2) < 0 \\ b^1 - c^1 < 0 \\ c^2 - b^2 < 0 \\ (1 - b^1 - b^2) - (1 - c^1 - c^2) < 0 \\ c^1 - a^1 < 0 \\ a^2 - c^2 < 0 \\ (1 - a^1 - a^2) - (1 - c^1 - c^2) < 0 \end{cases}$$

The corresponding matrix A_R is:

$$\begin{pmatrix} -1 & & 1 & & & \\ & 1 & & -1 & & \\ -1 & -1 & 1 & 1 & & \\ & & 1 & & -1 & \\ & & & -1 & & 1 \\ & & -1 & -1 & 1 & 1 \\ -1 & & & & 1 & \\ & 1 & & & & -1 \\ -1 & -1 & & & 1 & 1 \end{pmatrix} \begin{pmatrix} a^1 \\ a^2 \\ b^1 \\ b^2 \\ c^1 \\ c^2 \end{pmatrix} < 0$$

Proposition 5 *Let Δ be an abstract simplicial complex with vertex set V. Then Δ is a TD-Delaunay complex of H_d if and only if there exists a d-representation R of V such that $\Delta = \Sigma(R)$ and such that the corresponding TD-Delaunay system has a solution.*

Proof. ($\Rightarrow$) Consider a set of points $\mathcal{P}$ of H_d in general position and such that $\Delta = \tau(\mathcal{P})$. We can suppose that $V = \mathcal{P}$. Let $<_1, \dots, <_d$ be the d linear orders on $\mathcal{P}$ such that $u <_i v$ if $u^i < v^i$. Note that as the points are in general position these orders are well defined. Let R be the corresponding representation, and let us now check that $\Delta = \Sigma(R)$.

For any face $F \in \Delta$, there exists $c \in \mathbb{R}^d$ such that $\sum_{i=1}^d c^i \geq 1$ and such that $S = \{u \in H_d : u^1 \leq c^1, \dots, u^d \leq c^d\}$ contains exactly the points F, and they lie on its border. For every i, we note f_i the maximum of F in the order $<_i$. Suppose that $F \notin \Sigma(R)$. Then there exists a vertex z of $\mathcal{P}$ not in F such that z does not dominate F in any order of R. Thus $z <_i f_i$ for every i. Therefore $z^i < (f_i)^i \leq c^i$ for every i. Hence $z \in S$ contradicting the definition of S. Therefore $F \in \Sigma(R)$ and $\Delta \subseteq \Sigma(R)$.

Consider any non-empty face $F \in \Sigma(R)$. For every i, we note f_i the maximum of F in the order $<_i$ and we define $c^i = (f_i)^i$. This defines $S = \{u \in H_d : u^1 \leq c^1, \dots, u^d \leq c^d\}$. First note that as $c^i \geq x^i$ for any point $x \in F$ we have that $\sum_{i=1}^d c^i \geq 1$. For every $x \in F$ there exists an i such that $x = f_i$. Therefore $x^i = c^i$ and $x^j \leq c^j$ for every $j \neq i$ (because $x \leq_j f_j$ and then $x^j \leq (f_j)^j$). Thus x lies on the border of S. For every $x \notin F$ there exists i such that $y <_i x$ for every $y \in F$. Therefore $c^i < x^i$ and $x \notin S$. Thus no point of $\mathcal{P}$ is in the interior of S and $S \cap \mathcal{P} = F$. Therefore $F \in \tau(\mathcal{P})$ and $\Sigma(R) \subseteq \Delta$.

Finally $\Delta = \tau(\mathcal{P}) = \Sigma(R)$ and note that the coordinates u^i gives a solution to the TD-Delaunay system of R. Indeed, for any $uv \in \Sigma(R)$, $u^i < v^i \Leftrightarrow u \leq_i v$.

($\Leftarrow$) Consider now a d-representation R of V such that the TD-Delaunay system of R has a solution, that are values $v^1, \dots, v^{d-1}$ for each $v \in V$. We now define the point set $\mathcal{P} = V$ by setting, for every $v \in V$, we define the point $(v^1, \dots, v^{d-1}, 1 - v^1 - \dots - v^{d-1})$ of H_d. If necessary, one can slightly perturb the position of the vertices in order to obtain points in general position that still fulfill the system constraints.

For any face $F \in \tau(\mathcal{P})$ there exists $c \in \mathbb{R}^d$ such that $\sum_{i=1}^d c^i \geq 1$ and such that $S = \{u \in H_d : u^1 \leq c^1, \dots, u^d \leq c^d\}$ contains exactly the points F, and they lie on its border. Suppose that $F \notin \Sigma(R)$. Then there exists $z \in V$ such that z does not dominate F in any order. We define f_i as the maximum of F in the order $<_i$ for every i. Then $z <_i f_i$ for every i. Therefore $z^i < (f_i)^i \leq c^i$ for every i. Thus z lies in the interior of S, a contradiction. Hence $F \in \Sigma(R)$ and $\tau(\mathcal{P}) \subseteq \Sigma(R)$.

Consider any non-empty face $F \in \Sigma(R)$. For every i, we note f_i the maximum of F in the order $<_i$ and we define $c^i = (f_i)^i$. This defines $S = \{u \in H_d : u^1 \leq c^1, \dots, u^d \leq c^d\}$. First note that for any vertex $x \in F$ and any i, we have that $c^i = (f_i)^i \geq x^i$ (as the edge $\{x, f_i\} \in \Sigma(R)$). We hence have that $\sum_{i=1}^d c^i \geq \sum_{i=1}^d x^i \geq 1$. For every $x \in F$ there exists an i such that $x = f_i$. Therefore $x^i = (f_i)^i = c^i$ and as $x^j \leq c^j$ for every j, we have that x is on the border of S. For every $x \notin F$ one can prove that there exists i and a path $(f_i = a_1, a_2, \dots, a_\ell = x)$ such that $a_j <_i a_{j+1}$ for every $j < \ell$. Therefore $c^i = (a_1)^i <$

$(a_2)^i < \ldots < (a_\ell)^i = x^i$ and $x \notin S$. Thus $F \in \tau(\mathcal{P})$ and $\Sigma(R) \subseteq \tau(\mathcal{P})$.

This concludes the proof of the proposition. □

Remark 6 *The previous proposition shows that a TD-Delaunay complex of H_d has Dushnik-Miller dimension d.*

The reciprocal statement is a natural question, as it holds for $d = 2$ and 3. It has been stated in [5] and in [6] (as an open problem).

Conjecture 7 ([5]) *For every d-representation R, the abstract simplicial complex $\Sigma(R)$ is a TD-Delaunay complex of H_d, that is a TD-Delaunay complex of the $(d-1)$-dimensional euclidean space.*

In [5] the author proves the conjecture when $\Sigma(R)$ already admits some particular embedding. In the following we show that actually this conjecture does not hold, already for $d = 4$.

3 Multi flows

Let G be an oriented graph of edge set E. A flow on G is a function of $f : E \to \mathbb{R}^+$. Let the divergence $\mathsf{div}(x)$ of a vertex x be given by $\mathsf{div}(x) = \sum_{y \in V:(y,x)\in E} f(y,x) - \sum_{y\in V:(x,y)\in E} f(x,y)$.

Definition 8 (Multi flow) *Let R be a d-representation of V. Let G be the 1-skeleton of $\Sigma(R)$. For every $i \in \{1,\ldots,d\}$ we define G_i as the oriented graph obtained from G by orienting each edge xy of G from x to y if and only if $x <_i y$.*

Let $\varphi_1, \ldots, \varphi_d$ be d flows respectively on each graph $G_1, \ldots, G_d$. Let $x \in V$, we define $\mathsf{div}_i(x)$ as the divergence of x according to the flow φ_i. A multi flow is a set of such flows satisfying the condition that $\forall i \in \{1,\ldots,d\}, \forall x \in V, \mathsf{div}_i(x) = \mathsf{div}_d(x)$.

A key ingredient for our main theorem is the celebrated Farkas Lemma.

Lemma 9 (Farkas' Lemma) *For any $m \times n$ real matrix A, either*

- *$Ax < 0$ admits a solution $x \in \mathbb{R}^m$, or*
- *${}^tAy = 0$ admits a nonzero solution $y \in (\mathbb{R}^+)^n$.*

Furthermore both cases are exclusive.

It leads to the following proposition, whose proof is only sketched due to lack of space.

Proposition 10 *Let R be a d-representation of a set V. Then either*

- *the corresponding TD-Delaunay system admits a solution, or*
- *R admits a nonzero multi flow (i.e. a multi flow with some $\varphi_i(uv) > 0$).*

Furthermore both cases are exclusive.

Proof. We only propose here a sketch of proof.

According to Farkas' Lemma, we only have to show that ${}^tA_R y = 0$ admits a nonzero solution in $(\mathbb{R}^+)^{md}$ (where m is the number of edges) is equivalent to have a nonzero multiflow on R.

Suppose that there exists a solution to ${}^tA_R y = 0$. Then y is indexed on the edges of $\Sigma(R)$ and on the d indexes. For an edge e and an indice i, we define $y_{e,i}$ as the corresponding coordinate.

For every $i \in \{1,\ldots,d\}$, we define the flow φ_i on G_i by $\varphi_i(e) = y_{e,i}$ for every edge e of $\Sigma(R)$. As $y_{e,i} \geq 0$ for every edge e and for every i, then φ_i are flows.

Furhtermore for every vertex x, for every indexes $i < d$,

$$\sum_{y:x\to y\in G_i} y_{(x,y),i} - \sum_{y:y\to x\in G_i} y_{(x,y),i} + \sum_{y:y\to x\in G_d} y_{(x,y),d} - \sum_{y:x\to y\in G_d} y_{(x,y),d} = 0$$

Therefore,

$$-\mathsf{div}_i(x) + \mathsf{div}_d(x) = 0$$

We conlcude that $\varphi_1, \ldots, \varphi_d$ is a multiflow on R.

Reciprocally, if there is a multiflow $\varphi_1, \ldots, \varphi_d$ on R, then we define $y_{e,i} = \varphi_i(e)$ for every i and every edge e of $\Sigma(R)$. By checking that ${}^tA_R y = 0$ we conclude the proof.

□

4 A counter-example to Conjecture 7

Theorem 11 *Let R be the following 4-representation of $\{1,2,3,4,5,6,7,8\}$:*

$<_1$	2	3	4	5	7	6	8	1
$<_2$	1	3	4	5	8	6	7	2
$<_3$	1	2	4	6	7	5	8	3
$<_4$	1	2	3	6	8	5	7	4

Then $\Sigma(R)$ is an abstract simplicial complex of Dushnik-Miller dimension 4 but it is not a TD-Delaunay complex of H_4.

Proof. One can show that any 4-representation R' of $\{1,2,3,4,5,6,7,8\}$ such that $\Sigma(R') = \Sigma(R)$ is equivalent to R up to permutations of the orders and up to a permutation of the smallest 3 elements in each order. This implies that for any such R' the oriented graphs $G_i[\{5,6,7,8\}]$ (i.e. the subgraph of G_i induced by the vertices 5, 6, 7 and 8) remain the same.

If $\Delta = \Sigma(R)$ is a TD-Delaunay complex of dimension 4, then according to Proposition 5 there exists

a 4-representation R' of V such that $\Sigma(R') = \Delta$ and such that the corresponding TD-Delaunay system admits a solution. According to Proposition 10 there is no multi flow for such R'.

However, there is a multi flow on R' that is depicted on Figure 4. Vertices $1, 2, 3, 4$ are not drawn as the flows on their incident edges are defined as null. The divergence of 5 and 6 is -1 and it is $+1$ for 7 and 8.

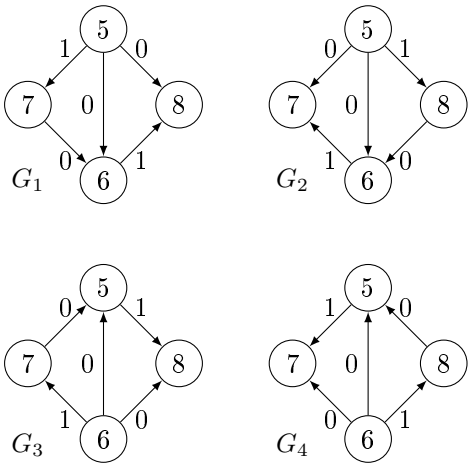


Figure 2: Definition of the multi flow on R'.

□

5 Conclusion

TD-Delaunay complexes have shown their interest in computational geometry since [3] as a very efficient tool to construct spanners in $\mathbb{R}^2$with small stretch [4]. It is likely that this extends to higher dimensional spaces. A question is for example whether graphs of Dushnik-Miller dimension 4 give better spanners than TD-Delaunay complexes in $\mathbb{R}^3$.

Furthermore, as the class of TD-Delaunay complexes of H_d is strictly included in the class of complexes of Dushnik-Miller dimension d, it may be interesting to study the problem of the grid embedding for this restricted class. Indeed Schnyder showed [2] that a planar graph with n vertices can be embedded in an $n \times n$ grid (without crossings). Ossona de Mendez showed [1] that a complex of Dushnik-Miller dimension $d + 1$ could be drawn in $\mathbb{R}^d$ without crossings. Nevertheless the last embedding uses exponentially large grids and it is still an open problem to reduce the sizes of these grids. Is it possible to reduce the size of these grids when dealing with TD-Delaunay complexes ?

6 Acknowledgments

The first author is thankful to Arnaud Mary and Vincent Pilaud for many discussions on this topic.

References

[1] P. Ossona de Mendez, Geometric Realization of Simplicial Complexes, *Proc. of Int. Symp. on Graph Drawing 1999* LNCS 1731, 323-332.

[2] W. Schnyder, Planar graphs and poset dimension, *Order* 5 (4): 323-343, 1989.

[3] P. Chew and R.L. Drysdale, Voronoi diagrams based on convex distance functions, *Proc. 1st Ann. Symp. on Computational Geometry*, pp 235-244, 1985.

[4] N. Bonichon, C. Gavoille, N. Hanusse, and D. Ilcinkas, Connections between theta-graphs, Delaunay triangulations, and orthogonal surfaces, *Proc. of WG '10*, LNCS 6410, 266-278, 2010.

[5] A. Mary, Dimension de poset et subdivisions simpliciales, Master Thesis, Univ. of Montpellier, 2010.

[6] W. Evans, S. Felsner, G. Kobourov, and T. Ueckerdt, Graphs admitting d-realizers: spanning-tree-decompositions and box-representations *Proc. of EuroCG 14*.

Star covering of red and blue points in the plane

Bernardo M. Ábrego* Silvia Fernández-Merchant† Mikio Kano‡ David Orden §
Pablo Pérez-Lantero¶ Carlos Seara ‖ Javier Tejel**

Abstract

A finite set of red and blue points in the plane in general position can be $K_{1,3}$-*covered* if the set can be partitioned into subsets of size 4 with 3 points of one color and 1 point of the other color in such a way that, if at each subset of four points we connect the uniquely-colored point to the other three points, then the resulting set of all segments has no crossings.

We consider the following problem: *Given a set R of r red points and a set B of b blue points in the plane in general position, how many points of $R \cup B$ can be $K_{1,3}$-covered*? and we prove the following results:

(1) If $r = 3g + h$ and $b = 3h + g$, for some nonnegative integers g and h, then there are point sets $R \cup B$, like $\{1, 3\}$-equitable sets (i.e., $r = 3b$ or $b = 3r$) and linearly separable sets, that can be $K_{1,3}$-covered.

(2) If $r = 3g+h$, $b = 3h+g$ and the points in $R \cup B$ are in convex position, then at least $r + b - 4$ points can be $K_{1,3}$-covered, and this bound is tight.

(3) There are arbitrarily large point sets $R \cup B$ in general position, with $|R| = |B|+1$, such that at most $r + b - 5$ points can be $K_{1,3}$-covered.

(4) If $b \leq r \leq 3b$, then at least $\frac{8}{9}(r + b - 8)$ points of $R \cup B$ can be $K_{1,3}$-covered. For $r > 3b$, at least $r - 3b$ red points will remain $K_{1,3}$-uncovered.

*bernardo.abrego@csun.edu. California State University, Northridge, USA.

†silvia.fernandez@csun.edu. California State University, Northridge, USA. Research supported by the NSF grant DMS-1400653.

‡mikio.kano.math@vc.ibaraki.ac.jp. Ibaraki University, Japan. Research supported by JSPS KAKENHI Grant Number 16K05248.

§david.orden@uah.es. Universidad de Alcalá, Spain. Research supported by MINECO Projects MTM2014-54207 and TIN2014-61627-EXP, TIGRE5-CM Comunidad de Madrid Project S2013/ICE-2919, and H2020-MSCA-RISE project 73499 - CONNECT.

¶pablo.perez.l@usach.cl. Universidad de Santiago, Chile. Research supported by CONICYT FONDECYT/Regular 1160543 (Chile), Millennium Nucleus Information and Coordination in Networks ICM/FIC RC130003 (Chile), and H2020-MSCA-RISE project 73499 - CONNECT.

‖carlos.seara@upc.edu. Universitat Politècnica de Catalunya, Spain. Research supported by projects Gen. Cat. DGR 2014SGR46, MINECO MTM2015-63791-R, and H2020-MSCA-RISE project 73499 - CONNECT.

**jtejel@unizar.es. Universidad de Zaragoza, Spain. Research supported by MINECO project MTM2015-63791-R, Gobierno de Aragón Grant E58 (ESF), and H2020-MSCA-RISE project 73499 - CONNECT.

1 Introduction

A large amount of research about discrete geometry on red and blue points in the plane has been done. More details can be found in the survey [3]. For given sets R of red points and B of blue points in the plane, so that $R \cup B$ is in general position (no three collinear points), we say that a graph G *covers* $R \cup B$ if the vertex set of G is $R \cup B$, every edge of G is a straight line segment connecting a red point and a blue point, and no two edges intersect except in their endpoints. Analogously, for a fixed graph G of k vertices, we say that $R \cup B$ has a *G-covering*, or can be *G-covered*, if $|R \cup B| = t \cdot k$ for some integer t, and the graph G_t resulting from the union of t copies of G covers $R \cup B$.

Some results on G-coverings are known. For example, it is well-known that, for a set $S = R \cup B$ of red and blue points in general position in the plane such that $|R| = |B|$, a *non-crossing geometric alternating perfect matching* always exists [3]. In other words, if the complete graph of order n is denoted by K_n, then S has a K_2-covering. The following theorem gives a sufficient condition for a given set of red and blue points to have a P_n-covering, where P_n denotes the (non-crossing alternating) path of length $2 \leq n \leq 12$.

Theorem 1 (Kaneko, Kano, and Suzuki [4]) *Let g and h be nonnegative integers. If n is an even integer such that $2 \leq n \leq 12$, then any set of $(n/2)g$ red points and $(n/2)g$ blue points in the plane in general position can be P_n-covered. If n is an odd integer such that $3 \leq n \leq 11$, then any set of $\lceil n/2 \rceil g + \lfloor n/2 \rfloor h$ red points and $\lfloor n/2 \rfloor g + \lceil n/2 \rceil h$ blue points in the plane in general position can be P_n-covered.*

Theorem 1 with $n = 3$ says that if $2g+h$ red points and $g+2h$ blue points are given in the plane in general position, then the points can be $K_{1,2}$-covered, where $K_{m,n}$ denotes the complete bipartite graph with stable parts of order m and n, respectively. $K_{1,n}$ is often called the *star* of order $n+1$, *centered* at the point in the stable part of order 1.

In this paper, we consider the following problem: *Given a set R of red points and a set B of blue points in the plane in general position, how many points of $R \cup B$ can be $K_{1,3}$-covered*? See Figure 1.

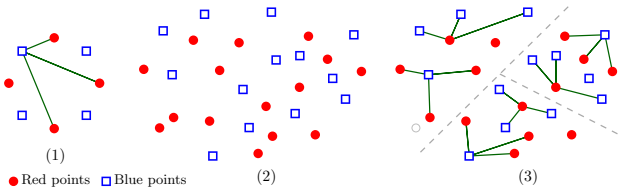


Figure 1: $K_{1,3}$-coverings in convex (1) and general (2)-(3) position.

For a set $S = R \cup B$ of red and blue points in general position, if R consists of exactly r red points and B consists of exactly b blue points, we say that S is an (r, b)-*set*. We define $\mathcal{C}(S)$ as the maximum number of points in S that can be $K_{1,3}$-covered. For nonnegative integers r and b, we define $\mathcal{C}(r, b)$ as the minimum of $\mathcal{C}(S)$ over all (r, b)-sets S in general position. Sometimes it is better to look at the number of points that are left *uncovered*. So we define $\mathcal{U}(S) = |S| - \mathcal{C}(S)$ and $\mathcal{U}(r, b) = r + b - \mathcal{C}(r, b)$. Then, $\mathcal{U}(r, b)$ is the maximum of $\mathcal{U}(S)$ over all (r, b)-sets S in general position.

Observe that, if an (r, b)-set S admits a $K_{1,3}$-covering consisting of $h \geq 0$ stars centered in red points and $g \geq 0$ stars centered in blue points, then $r = 3g + h$, $b = 3h + g$ and then $|S| = 4(g + h)$. But this condition on the number of red and blue points is not sufficient to assure that a $K_{1,3}$-covering exists.

2 Particular configurations

2.1 Equitable sets

Let S be an (r, b)-set such that either $r = 3b$ or $b = 3r$. In any of both cases we say that S is a $\{1, 3\}$-*equitable set*. The following theorem, which is a generalization of the Ham-sandwich Theorem, allows us to show that any $\{1, 3\}$-equitable set can be $K_{1,3}$-covered.

Theorem 2 (Equitable Subdivision [1, 2, 5]) *Let c, d and g be positive integers. If cg red points and dg blue points are given in the plane in general position, then there exists a subdivision of the plane into g convex regions such that each region contains precisely c red points and d blue points.*

Theorem 3 *If an (r, b)-set S is $\{1, 3\}$-equitable, then all the points of S can be $K_{1,3}$-covered.*

Proof. W.l.o.g., assume $r = 3b$. Apply directly Theorem 2 with $c = 3$, $d = 1$, and $g = b$. The points in each region can be trivially $K_{1,3}$-covered and the union of these coverings is a $K_{1,3}$-covering of S. □

2.2 Linearly separable sets

An (r, b)-set $S = R \cup B$ is *linearly separable* if there exists a line ℓ that separates R and B. For $r = 3g + h$ and $b = 3h + g$, we have the following result.

Theorem 4 *If $S = R \cup B$ is a linearly separable $(3g + h, 3h + g)$-set, then all the points of S can be $K_{1,3}$-covered.*

Proof. W.l.o.g., assume $h > 0$. The proof is by induction on g. If $g = 0$, then S is a $\{1, 3\}$-equitable set, which can be $K_{1,3}$-covered by Theorem 3. Assume $g > 0$, and suppose R and B are separable by the line ℓ. Then, there are lines ℓ_b and ℓ_r parallel to ℓ such that ℓ_b separates 4 blue points from the rest of S and ℓ_r separates 4 red points from the rest of S. See Figure 2. For these ℓ_b and ℓ_r it can be proved, by continuous rotation, that there is a line ℓ' separating 4 points of S, exactly 3 of them red, from the rest of S. This set of 4 points can clearly be $K_{1,3}$-covered. The rest of S is a $(3(g-1)+h, 3h+(g-1))$-set, still linearly separable, and thus – since ℓ' separates the relevant four points from S – it can also be $K_{1,3}$-covered by induction. The two covers are separated by ℓ' and thus they together form a $K_{1,3}$-cover of S. □

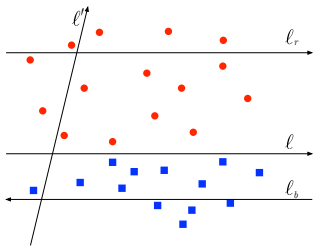


Figure 2: Illustration for the proof of Theorem 4.

Remark 1 *Theorems 3 and 4 can be extended to, given $k \geq 4$, obtain $K_{1,k}$-coverings of $(kg + h, kh + g)$-sets, where g and h are nonnegative integers.*

2.3 Convex position

We have shown some $(3g+h, 3h+g)$-sets which always admit a $K_{1,3}$-covering. Now we show that this is not the case for some other $(3g + h, 3h + g)$-sets.

Theorem 5 *If $S = R \cup B$ is a $(3g + h, 3h + g)$-set in convex position, then at least $4(g + h) - 4$ points of S can be $K_{1,3}$-covered and this bound is tight.*

Proof. The proof is by induction on $|R \cup B|$. If $g = 0$ or $h = 0$, which includes the base case $|R \cup B| = 4$, then S is a $\{1, 3\}$-equitable set, which can be $K_{1,3}$-covered by Theorem 3. Thus, we assume $g, h > 0$, $|R \cup B| \geq 8$ and the elements of $R \cup B$ are ordered clockwise along the boundary $\partial \operatorname{conv}(R \cup B)$.

1. Suppose that there exists a set $X \subset S$ of four consecutive points cyclically on $\partial \operatorname{conv}(S)$ such that 3 of them have the same color and the remaining one has a distinct color. Then, the four points of X can be

$K_{1,3}$-covered, and $4(g+h-1)-4$ points of $(R\cup B)\setminus X$ (which is either a $(3g+(h-1),3(h-1)+g)$-set or a $(3(g-1)+h,3h+(g-1))$-set) can be $K_{1,3}$-covered because of the induction hypothesis. Note that, by convexity, these two $K_{1,3}$-coverings do not cross, so the desired covering is obtained.

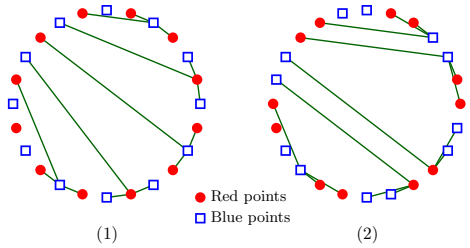


Figure 3: Illustration for the proof of Theorem 5.

2. Suppose that such a set X does not exist. By a simple counting argument, this implies one of these two cases: One red point and one blue point alternately lie on $\partial\,\mathrm{conv}(S)$, or two red points and two blue points alternately lie on $\partial\,\mathrm{conv}(S)$ (see Figure 3). Therefore, in both cases, $|R|=|B|$, $g=h$, $|R|=4g=|B|$, and $|S|=|R|+|B|=8g$. Assume that the $n=8g$ points of S are numbered from 1 to n clockwise around $\partial\,\mathrm{conv}(S)$.

(i) If one red point and one blue point alternate on $\partial\,\mathrm{conv}(R\cup B)$, we obtain the desired $K_{1,3}$-covering as follows: For $k=1,\ldots,2g-1$, we take the points with numbers $3k-2$, $3k-1$, $3k$, and $n-k$ and cover them with a $K_{1,3}$ (see Figure 3.(1)). Doing this, we leave uncovered precisely the four points numbered $6g-2$, $6g-1$, $6g$, and n, respectively.

(ii) If two red points and two blue points alternate on $\partial\,\mathrm{conv}(R\cup B)$, we can obtain the desired $K_{1,3}$-covering as follows: For $k=1,\ldots,2g-1$, we take the points with numbers $3k-2$, $3k-1$, $3k$, and $n-k-1$ and cover them with a $K_{1,3}$ (see Figure 3.(2)). Doing this, we leave uncovered precisely the four points numbered $6g-2$, $6g-1$, $n-1$, and n, respectively.

Finally, the fact that the bound $4(g+h)-4$ is the best possible can be proved later as a consequence of part (b) of Theorem 6 (presented below), because we are covering with $K_{1,3}$ stars. □

2.4 Some (r,b)-sets with $\mathcal{U}(r,b)>0$

We give some particular configurations of (r,b)-sets for which $\mathcal{U}(r,b)\geq 5$ and prove that, except for $\{1,3\}$-equitable sets, $\mathcal{U}(r,b)>0$ for any values of r and b.

Theorem 6 *Let r and b be nonnegative integers. (a) $\mathcal{U}(r,b)=0$ if and only if $r=3b$ or $b=3r$. (b) If $r=b$, the (r,b)-set in convex position where the blue and red points alternate along the convex hull cannot be completely $K_{1,3}$-covered. (c) $\mathcal{U}(2k+1,2k)\geq 5$ for any $k\geq 4$.*

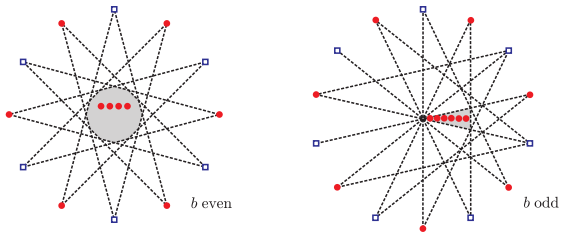


Figure 4: Cases b even and b odd in Theorem 6(a).

Proof. (a) Assume that $r\geq b$ and $r\neq 3b$. If $r>3b$, a perfect covering of any (r,b)-set is numerically impossible as there are too many red points. If $r<3b$, the following (r,b)-set S cannot completely be $K_{1,3}$-covered: Draw a color-alternating convex $2b$-gon together with a set Q of $r-b$ almost collinear red points as illustrated in Figure 4. The set Q is not split by any bichromatic diagonal and its points are almost along a line ℓ through the midpoints of two antipodal sides. Note that in a perfect $K_{1,3}$-covering there must be at least one star T with red center, since $r<3b$. If this center of T is in Q, then T must have at least two consecutive edges on the same side of ℓ (see Figure 5, left). But these two edges isolate an odd number of points in S, and therefore these points cannot be $K_{1,3}$-covered. If the red center of T is not in Q, then the set Q must be either to the left or to the right of at least two edges of T (see Figure 5, right). As before, two consecutive such edges isolate an odd number of points in S from the rest and so they cannot be completely $K_{1,3}$-covered. If $r=3b$, then Theorem 3 guarantees $\mathcal{U}(r,b)=0$.

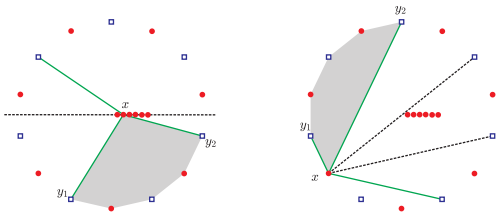


Figure 5: Red center in Q or on the convex hull.

(b) See Figure 3, left. A perfect covering would need a star with red center. But the region delimited by two consecutive edges of such a star would isolate an odd number of points, which cannot be $K_{1,3}$-covered.

(c) Consider the $(2k+1,2k)$-set S formed by the color alternating vertices of a regular $4k$-gon and one red point x near the center of the polygon. Since $\mathcal{U}(2k+1,2k)\equiv 1 \pmod 4$, it is enough to show that $\mathcal{U}(S)>1$. Note that in a $K_{1,3}$-covering of S leaving only one point uncovered, there must be exactly $\lfloor k/2\rfloor\geq 2$ stars with red center. If there is a star T whose center is x, each of the sectors determined by the edges of T contains an odd number of points

(see Figure 6, left). Thus, each of these subsets of S cannot be completely $K_{1,3}$-covered.

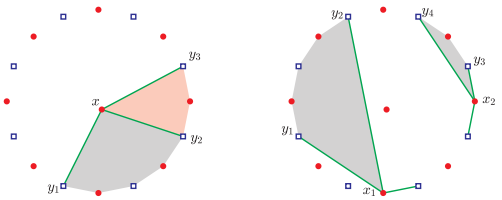


Figure 6: Red center in Q or on the convex hull.

If there are two stars with red centers that are not x, then each star has two consecutive edges (diagonals) delimiting a region that does not contain x and thus isolating an odd number of points that cannot completely be $K_{1,3}$-covered (see Figure 6, right). In either case there are at least two points not covered. □

3 General configurations of points

Given an (r, b)-set S in general position, the main goal is to give a lower bound on the number of points of S that can always be $K_{1,3}$-covered. Let r and b be positive integers and $\alpha = \frac{b}{r}$. Assume that $b \leq r$. Notice that, if $\alpha < 1/3$, then $r > 3b$ and there are too many red points to be covered by stars centered in blue points. In this case, at least $r - 3b$ red points will necessarily remain uncovered in any $K_{1,3}$-covering. Moreover, the bound is tight because, by removing $r - 3b$ red points, we obtain an $\{1, 3\}$-equitable set that can be $K_{1,3}$-covered by Theorem 3. When $1/3 \leq \alpha \leq 1$, we prove that at least $\frac{8}{9}(r+b-8)$ points can be always $K_{1,3}$-covered.

3.1 Lower bound when $1/3 \leq \alpha \leq 4/5$

Theorem 7 *Let t be a nonnegative integer. Then, $\mathcal{U}(3k - t, k + 2t) \leq t$ for any integer $k \geq \frac{5}{8}t$.*

This result (proof omitted due to lack of space) implies the following lower bound on $\mathcal{C}(r, b)$.

Theorem 8 *If $\frac{1}{3} \leq \alpha \leq 1$, then*

$$\mathcal{C}(r, b) \geq \frac{4}{7}\left(\frac{\alpha + 2}{\alpha + 1}\right)(r + b) - 4.$$

Corollary 1 *If $\frac{1}{3} \leq \alpha \leq \frac{4}{5}$, then at least $\frac{8}{9}(r+b)-4$ points can be $K_{1,3}$-covered.*

3.2 Lower bound when $4/5 \leq \alpha \leq 1$

Theorem 9 (Kaneko, Kano and Suzuki [4]) *Let $s \geq 1$, $g \geq 0$ and $h \geq 0$ be integers such that $g + h \geq 1$. Assume that $|R| = (s + 1)g + sh$ and $|B| = sg + (s + 1)h$. Then, there exists a subdivision $X_1 \cup \cdots \cup X_g \cup Y_1 \cup \cdots \cup Y_h$ of the plane into $g + h$ disjoint convex regions such that every X_i contains exactly $s + 1$ red points and s blue points and every Y_j contains exactly s red points and $s+1$ blue points.*

Theorem 10 *If $\frac{4}{5} \leq \alpha \leq 1$, $\mathcal{C}(r, b) \geq \frac{8}{9}(r + b - 8)$.*

Proof. Assume $r + b \geq 9$, otherwise there is nothing to prove. If $\frac{4}{5} \leq \alpha \leq 1$, then $4r \leq 5b \leq 5r$. Write the nonnegative integer $5b - 4r = 9n + m$ for nonnegative integers n and m, $0 \leq m \leq 8$. Define

$$(h, g) = \begin{cases} (n, r - b + n) & \text{if } 0 \leq m \leq 4 \\ (n + 1, r - b + n) & \text{if } 5 \leq m \leq 8 \end{cases}.$$

Observe that if $0 \leq m \leq 4$, then $5h + 4g = b - m$ and $4h + 5g = r - m$; and if $5 \leq m \leq 8$, then $5h + 4g = b - (m - 5)$ and $4h + 5g = r - (m - 4)$. In either case, after removing m blue points and m red points (in the first case) or $m - 5$ blue points and $m - 4$ red points (in the second case), we end up with a set having $5h+4g$ blue points and $4h+5g$ red points. According to Theorem 9 with $s = 4$ (note that $h + g \geq 1$ since $r + b \geq 9$), the plane can be partitioned into $g + h$ disjoint convex regions $X_1 \cup \cdots \cup X_g \cup Y_1 \cup \cdots \cup Y_h$ such that every X_i contains exactly 5 red points and 4 blue points, and every Y_j contains exactly 4 red points and 5 blue points. Finally, by Theorem 7 (with $k = 2$ and $t = 1$), eight points in every X_i and Y_j can be $K_{1,3}$-covered. Thus $\mathcal{C}(r, b) \geq \mathcal{C}(4h + 5g, 5h + 4g) \geq 8(h + g) \geq \frac{8}{9}(r + b - 8)$. □

Open problem: Is there a red-blue point configuration with $b \leq r \leq 3b$ such that $\omega(1)$ points are necessarily left $K_{1,3}$-uncovered?

References

[1] S. Bespamyatnikh, D. Kirkpatrick, and J. Snoeyink. Generalizing ham sandwich cuts to equitable subdivisions. *Discrete & Computational Geometry*, Vol. 24(4), (2000), 605–622.

[2] H. Ito, H. Uehara, and M. Yokoyama. 2-Dimension ham sandwich theorem for partitioning into three convex pieces. *Lecture Notes in Computer Science*, Vol. 1763, (2000), 129–157.

[3] A. Kaneko and M. Kano. Discrete geometry on red and blue points in the plane: A survey. *Discrete & Computational Geometry*, Vol. 25 of Algorithms and Combinatorics, (2003), 551–570.

[4] A. Kaneko, M. Kano, and K. Suzuki. Path coverings of two sets of points in the plane. *Contemporary Mathematics of AMS*, Vol. 342, (2004), 99–112.

[5] T. Sakai. Balanced Convex Partitions of Measures in $\mathbb{R}^2$. *Graphs and Combinatorics*, Vol. 18(1), (2002), 169–192.

Bounds on the angle between tangent spaces and the metric distortion for C^2 manifolds with given positive reach.

Mathijs Wintraecken*

1 Introduction

In this paper we discuss two results: Firstly, we provide a tight bound on difference between the Euclidean distance and geodesic distance between two points p and q on a C^2 manifold $\mathcal{M}$ embedded in $\mathbb{R}^d$. Secondly, we give tight bounds on the angles between the tangent spaces of two points on $\mathcal{M}$. In both cases we assume that p and q are not too far from each other, where far can be quantified in terms of the reach of the manifold.

Our exposition is the result of combining the work of Niyogi, Smale, and Weinberger [NSW08], and the two dimensional analysis of Attali, Edelsbrunner, and Mileyko [AEM07] with some observations concerning the reach. The reach was introduced in the seminal work of Federer [Fed59], of which the authors of [NSW08] seem to have been unaware. Versions of all results that we present here, but often in a weaker form, can be found in [NSW08]. We would like to stress that some effort went into simplifying the explanation, in particular of the second fundamental form found in that article.

A significant amount of attention has gone to bounds like the ones we described above, see [CDR05, DGRS08, BG10, BDG13] to name but a few, which hopefully justifies the writing of this exposition.

1.1 Definitions and notation

We assume that $\mathcal{M}$ is an n-dimensional smooth (by which we mean C^2) manifold without boundary, embedded in $\mathbb{R}^d$. The tangent bundle is denoted by $T\mathcal{M}$ and the tangent plane at the point $p \in \mathcal{M}$ by $T_p\mathcal{M}$, which we shall often regard as embedded in $\mathbb{R}^d$ without explicitly refering to the embedding. The normal bundle will be denoted by $N\mathcal{M}$ and the normal space at a point $p \in \mathcal{M}$ by $N_p\mathcal{M}$.

Geodesic distances on manifolds between $p, q \in \mathcal{M}$ are denoted by $d_{\mathcal{M}}(p, q)$. The distance between the points x and y in Euclidean space is denoted by $|x-y|$.

For an embedded manifold, the medial axis ($\mathrm{ax}(\mathcal{M})$) is the set of points in the ambient space for which there are at least two points on the manifold that attain the minimal distance to the point in ambient space. The reach of $\mathcal{M}$ is the minimal distance from the medial axis to the manifold and is denoted by $\mathrm{rch}(\mathcal{M})$.

Let $N(\mathcal{M}, r) = \{x \in \mathbb{R}^d \mid d_{\mathbb{E}}(x, \mathcal{M}) < r\}$ be a neighbourhood of radius $r < \mathrm{rch}(\mathcal{M})$. The projection of a point $x \in N(\mathcal{M}, r)$ onto the closest point on $\mathcal{M}$ will be denoted by $\pi_{\mathcal{M}}$.

1.2 Federer's results

We shall be using a corollary of the following result due to Federer, [Fed59, Theorem 4.8 (12)]:

Lemma 1 (Federer's tubular neighbourhoods) *Let $B_{N_p\mathcal{M}}(r)$, be the $d-n$ dimensional ball of radius r centred at p in the normal space $N_p\mathcal{M}$, where $r < \mathrm{rch}\,\mathcal{M}$. We emphasize that we see $N_p\mathcal{M}$ as a subspace of $\mathbb{R}^d$. For every point $x \in B_{N_p\mathcal{M}}(r)$, $\pi_{\mathcal{M}}(x) = p$.*

From this we immediately see that:

Corollary 2 *Let $\mathcal{M}$ be a submanifold of $\mathbb{R}^d$ and $p \in \mathcal{M}$. Any open ball $B(c, r)$ that is tangent to $\mathcal{M}$ at p and whose radius r satisfies $r \leq \mathrm{rch}(M)$ does not intersect $\mathcal{M}$.*

Proof. Let $r < \mathrm{rch}(M)$. Suppose that the intersection of $\mathcal{M}$ and the open ball is not empty, then the $\pi_{\mathcal{M}}(c) \neq p$ contradicting Federer's result. The result for $r = \mathrm{rch}(M)$ now follows by taking the limit. □

2 Angle and distance distortion bounds

We first consider the metric distortion between the Euclidean and geodesic distance between two points on a manifold. For this we need the following lemma:

Lemma 3 *Let $\gamma(t)$ be a geodesic parametrized according to arc length on $\mathcal{M} \subset \mathbb{R}^d$, then $|\ddot{\gamma}| \leq 1/\mathrm{rch}(\mathcal{M})$, where we use Newton's notation.*

Proof. Because $\gamma(t)$ is a geodesic, $\ddot{\gamma}(t)$ is normal to $\mathcal{M}$ at $\gamma(t)$. Now consider the sphere of radius $\mathrm{rch}(\mathcal{M})$ tangent to $\mathcal{M}$ at $\gamma(t)$, whose centre lies on the line $\{\gamma(t) + \lambda\ddot{\gamma}(t) \mid \lambda \in \mathbb{R}\}$. If now $|\ddot{\gamma}|$ were larger than $1/\mathrm{rch}(\mathcal{M})$, the geodesic γ would enter the sphere tangent to $\mathcal{M}$ at $\gamma(t)$, which would contradict Corollary 2. □

*INRIA Sophia-Antipolis

We now have the following, which is the straightforward generalization of Property I of [AEM07] to arbitrary dimension and using the reach:

Lemma 4 *Let $p, q \in \mathcal{M}$ be such that $d_{\mathcal{M}}(p,q) \leq \pi \mathrm{rch}(\mathcal{M})$, then*

$$2\,\mathrm{rch}(\mathcal{M}) \sin\left(\frac{d_{\mathcal{M}}(p,q)}{2\mathrm{rch}(\mathcal{M})}\right) \leq |p-q|.$$

Proof. Let γ be a geodesic parametrized according to arc length whose length ℓ therefore equals $d_{\mathcal{M}}(p,q)$, such that $\gamma(0) = p$ and $\gamma(\ell) = q$. Because γ is parametrized according to arc length $|\dot\gamma| = 1$ and $\dot\gamma(t)$ can be seen as a curve on the sphere $\mathbb{S}^{d-1}$. Moreover $\ddot\gamma$ can be seen as tangent to this sphere. The angle between two tangent vectors $\dot\gamma(a)$ and $\dot\gamma(b)$ equals the geodesic distance on the sphere. The geodesic distance between any two points is smaller or equal to the length of any curve connecting these points, and $\{\dot\gamma(t) \mid t \in [a,b]\}$ is such a curve. We therefore have

$$\angle\dot\gamma(a)\dot\gamma(b) \leq \int_a^b \left|\frac{d}{dt}\dot\gamma\right| \mathrm{d}t = \int_a^b |\ddot\gamma| \mathrm{d}t \leq \frac{|a-b|}{\mathrm{rch}(\mathcal{M})},$$

where we used Lemma 3.

We now write $T = \dot\gamma(\frac{\ell}{2})$. The length of γ in the direction of T is

$$\begin{aligned}
\langle q-p, T\rangle &= \int_0^\ell \langle\dot\gamma(s), T\rangle \mathrm{d}s \\
&= \int_0^{\ell/2} \langle\dot\gamma(s), T\rangle \mathrm{d}s + \int_{\ell/2}^{\ell} \langle\dot\gamma(s), T\rangle \mathrm{d}s \\
&\geq \int_0^{\ell/2} \cos\frac{|s-\ell/2|}{\mathrm{rch}(\mathcal{M})} \mathrm{d}s + \int_{\ell/2}^{\ell} \cos\frac{|s-\ell/2|}{\mathrm{rch}(\mathcal{M})} \mathrm{d}s \\
&= 2\,\mathrm{rch}(\mathcal{M}) \sin\left(\frac{\ell}{2\mathrm{rch}(\mathcal{M})}\right).
\end{aligned}$$

Because $|q-p| \geq \langle q-p, T\rangle$, the result follows. □

This bound is tight as it is attained on the sphere of the appropriate dimension.

2.1 Angles between tangent spaces

We can now turn our attention to the angles between tangent spaces at different points of the manifold. Here we mainly follow Niyogi, Smale, and Weinberger [NSW08], but use one useful observation of [AEM07]. We shall be using the second fundamental form, which we assume the reader to be familiar with. We refer to [dC92] as a standard reference.

The second fundamental form $\mathrm{I\!I}_p(u,v)$ has the geometric interpretation of the normal part of the covariant derivative, where we assume now that u, v are vector fields. In particular $\mathrm{I\!I}(u,v) = \bar\nabla_u v - \nabla_u v$, where $\bar\nabla$ is the connection in the ambient space, in this case Euclidean space, and ∇ the connection with respect to the induced metric on the manifold $\mathcal{M}$. $\mathrm{I\!I}_p : T_p\mathcal{M} \times T_p\mathcal{M} \to N_p\mathcal{M}$ is a symmetric bi-linear form, see for example Section 6.2 of [dC92] for a proof. This means that we only need to consider vectors in the tangent space and not vector fields, when we consider $\mathrm{I\!I}_p(u,v)$.

We can now restrict our attention to u, v lying on the unit sphere $\mathbb{S}^{n-1}_{T_p\mathcal{M}}$ in the tangent space and ask for which of these vectors $|\mathrm{I\!I}_p(u,v)|$ is maximized. Let us assume that the $\mathrm{I\!I}_p(u,v)$ for which the maximum[1] is attained lies in the direction of $\eta \in N_p\mathcal{M}$ where η is assumed to have unit length.

We can now identify $\langle\mathrm{I\!I}_p(\cdot,\cdot),\eta\rangle$, with a symmetric matrix.[2] Because of this $\langle\mathrm{I\!I}_p(u,v),\eta\rangle$, with $u, v \in \mathbb{S}^{n-1}_{T_p\mathcal{M}}$, attains its maximum for u, v both lying in the direction of the unit eigenvector w of $\langle\mathrm{I\!I}_p(\cdot,\cdot),\eta\rangle$ with the largest[3] eigenvalue. In other words the maximum is assumed for $u = v = w$. Let us now consider a geodesic γ_w on $\mathcal{M}$ parametrized by arclength such that $\gamma_w(0) = p$ and $\dot\gamma_w(0) = w$. Now, because γ_w is a geodesic and the ambient space is Euclidean,

$$\begin{aligned}
\mathrm{I\!I}_p(w,w) &= \mathrm{I\!I}_p(\dot\gamma_w, \dot\gamma_w) \\
&= \bar\nabla_{\dot\gamma_w}\dot\gamma_w - \nabla_{\dot\gamma_w}\dot\gamma_w \\
&= \bar\nabla_{\dot\gamma_w}\dot\gamma_w - 0 \\
&= \ddot\gamma_w.
\end{aligned}$$

Due to Lemma 3 and by definition of the maximum, we now see that $|\mathrm{I\!I}_p(u,v)| \leq |\mathrm{I\!I}_p(w,w)| \leq 1/\mathrm{rch}\mathcal{M}$, for all u, v of length one.

Having discussed the second fundamental form, we can give the following lemma:

Lemma 5 *Let $p, q \in \mathcal{M}$, then*

$$\angle T_p\mathcal{M} T_q\mathcal{M} \leq \frac{d_{\mathcal{M}}(p,q)}{\mathrm{rch}(\mathcal{M})}.$$

Proof. As in the previous lemma γ is a geodesic connecting p and q, parametrized in the same manner. We consider an arbitrary unit vector u and parallel transport this unit vector along γ, getting the unit vectors $u(t)$ in the tangent spaces $T_{\gamma(t)}\mathcal{M}$. The maximal angle between $u(0)$ and $u(\ell)$, for all u bounds the angle between $T_p\mathcal{M}$ and $T_q\mathcal{M}$. Now

$$\begin{aligned}
\frac{du}{dt} &= \bar\nabla_{\dot\gamma} u(t) \\
&= \mathrm{I\!I}_p(\dot\gamma, u(t)) + \nabla_{\dot\gamma} u(t) \\
&= \mathrm{I\!I}_p(\dot\gamma, u(t)) + 0,
\end{aligned}$$

where we used that $u(t)$ is parallel and thus by definition $\nabla_{\dot\gamma} u(t) = 0$. So using our discussion above

[1] If there is more than one direction we simply pick one.
[2] We assume we transpose the first vector.
[3] We can assume positivity without loss of generality, and, again, if there is more than one direction, we pick one.

$|\frac{du}{dt}| \leq 1/\mathrm{rch}(\mathcal{M})$. Now we note that, similarly to what we have seen in the proof of Lemma 4, $u(t)$ can be seen as a curve on the sphere and thus $\angle u(0)u(\ell) \leq \int_0^\ell |\frac{du}{dt}| \mathrm{d}t \leq \ell/\mathrm{rch}(\mathcal{M})$. □

This bound is tight as it is again attained for a sphere.

Combining Lemmas 4 and 5 we find that

Corollary 6 *Under the same conditions as Lemma 4:*

$$\sin\left(\frac{\angle T_p\mathcal{M}T_q\mathcal{M}}{2}\right) \leq \frac{|p-q|}{2\mathrm{rch}(\mathcal{M})}.$$

Proof. Lemma 5 gives

$$\sin\left(\frac{\angle T_p\mathcal{M}T_q\mathcal{M}}{2}\right) \leq \sin\left(\frac{d_{\mathcal{M}}(p,q)}{2\mathrm{rch}(\mathcal{M})}\right)$$

and Lemma 4 yields

$$\sin\left(\frac{d_{\mathcal{M}}(p,q)}{2\mathrm{rch}(\mathcal{M})}\right) \leq \frac{|p-q|}{2\mathrm{rch}(\mathcal{M})}.$$

The result now follows. □

Remark 7 *Although the results were given in terms of the (global) reach to simplify the exposition, the results can be easily generalized to accommodate the local geometry.*

Acknowledgements

The work presented here is part of a larger ongoing project involving Jean-Daniel Boissonnat, André Lieutier, and Mael Rouxel-Labbé, and their collaboration is gratefully acknowledged. I would also like to thank all other members of the Datashape team (previously known as Geometrica). The research leading to these results has received funding from the European Research Council (ERC) under the European Union's Seventh Framework Programme (FP/2007-2013)/ERC Grant Agreement No. 339025 GUDHI (Algorithmic Foundations of Geometry Understanding in Higher Dimensions).

References

[AEM07] Dominique Attali, Herbert Edelsbrunner, and Yuriy Mileyko. Weak Witnesses for Delaunay triangulations of Submanifold. In *ACM Symposium on Solid and Physical Modeling*, pages 143–150, Beijing, China, June 2007.

[BDG13] J.-D. Boissonnat, R. Dyer, and A. Ghosh. Constructing intrinsic Delaunay triangulations of submanifolds. Research Report RR-8273, INRIA, 2013. arXiv:1303.6493.

[BG10] J.-D. Boissonnat and A. Ghosh. Triangulating smooth submanifolds with light scaffolding. *Mathematics in Computer Science*, 4(4):431–461, 2010.

[CDR05] S.-W. Cheng, T. K. Dey, and E. A. Ramos. Manifold reconstruction from point samples. In *SODA*, pages 1018–1027, 2005.

[dC92] M. P. do Carmo. *Riemannian Geometry*. Birkhäuser, 1992.

[DGRS08] T.K. Dey, J. Giesen, E.A. Ramos, and B. Sadri. Critical points of distance to an ϵ-sampling of a surface and flow-complex-based surface reconstruction. *International Journal of Computational Geometry & Applications*, 18(01n02):29–61, 2008.

[Fed59] H. Federer. Curvature measures. *Trans. Amer. Math. Soc.*, 93(3):418–491, 1959.

[NSW08] P. Niyogi, S. Smale, and S. Weinberger. Finding the homology of submanifolds with high confidence from random samples. *Discrete & Comp. Geom.*, 39(1-3):419–441, 2008.

$|\frac{du}{dt}| \leq 1/\mathrm{rch}(\mathcal{M})$. Now we note that, similarly to what we have seen in the proof of Lemma 4, $u(t)$ can be seen as a curve on the sphere and thus $\angle u(0)u(\ell) \leq \int_0^\ell |\frac{du}{dt}| \mathrm{d}t \leq \ell/\mathrm{rch}(\mathcal{M})$. □

This bound is tight as it is again attained for a sphere.

Combining Lemmas 4 and 5 we find that

Corollary 6 *Under the same conditions as Lemma 4:*

$$\sin\left(\frac{\angle T_p\mathcal{M}T_q\mathcal{M}}{2}\right) \leq \frac{|p-q|}{2\mathrm{rch}(\mathcal{M})}.$$

Proof. Lemma 5 gives

$$\sin\left(\frac{\angle T_p\mathcal{M}T_q\mathcal{M}}{2}\right) \leq \sin\left(\frac{d_{\mathcal{M}}(p,q)}{2\mathrm{rch}(\mathcal{M})}\right)$$

and Lemma 4 yields

$$\sin\left(\frac{d_{\mathcal{M}}(p,q)}{2\mathrm{rch}(\mathcal{M})}\right) \leq \frac{|p-q|}{2\mathrm{rch}(\mathcal{M})}.$$

The result now follows. □

Remark 7 *Although the results were given in terms of the (global) reach to simplify the exposition, the results can be easily generalized to accommodate the local geometry.*

Acknowledgements

The work presented here is part of a larger ongoing project involving Jean-Daniel Boissonnat, André Lieutier, and Mael Rouxel-Labbé, and their collaboration is gratefully acknowledged. I would also like to thank all other members of the Datashape team (previously known as Geometrica). The research leading to these results has received funding from the European Research Council (ERC) under the European Union's Seventh Framework Programme (FP/2007-2013)/ERC Grant Agreement No. 339025 GUDHI (Algorithmic Foundations of Geometry Understanding in Higher Dimensions).

References

[AEM07] Dominique Attali, Herbert Edelsbrunner, and Yuriy Mileyko. Weak Witnesses for Delaunay triangulations of Submanifold. In *ACM Symposium on Solid and Physical Modeling*, pages 143–150, Beijing, China, June 2007.

[BDG13] J.-D. Boissonnat, R. Dyer, and A. Ghosh. Constructing intrinsic Delaunay triangulations of submanifolds. Research Report RR-8273, INRIA, 2013. arXiv:1303.6493.

[BG10] J.-D. Boissonnat and A. Ghosh. Triangulating smooth submanifolds with light scaffolding. *Mathematics in Computer Science*, 4(4):431–461, 2010.

[CDR05] S.-W. Cheng, T. K. Dey, and E. A. Ramos. Manifold reconstruction from point samples. In *SODA*, pages 1018–1027, 2005.

[dC92] M. P. do Carmo. *Riemannian Geometry*. Birkhäuser, 1992.

[DGRS08] T.K. Dey, J. Giesen, E.A. Ramos, and B. Sadri. Critical points of distance to an ϵ-sampling of a surface and flow-complex-based surface reconstruction. *International Journal of Computational Geometry & Applications*, 18(01n02):29–61, 2008.

[Fed59] H. Federer. Curvature measures. *Trans. Amer. Math. Soc.*, 93(3):418–491, 1959.

[NSW08] P. Niyogi, S. Smale, and S. Weinberger. Finding the homology of submanifolds with high confidence from random samples. *Discrete & Comp. Geom.*, 39(1-3):419–441, 2008.

Near-Optimal ε-Kernel Construction and Related Problems

Sunil Arya* Guilherme D. da Fonseca† David M. Mount‡

Abstract

The computation of (i) ε-kernels, (ii) approximate diameter, and (iii) approximate bichromatic closest pair are fundamental problems in geometric approximation. In each case the input is a set of points in $\mathbb{R}^d$ for a constant dimension d and an approximation parameter $\varepsilon > 0$. In this paper, we describe new algorithms for these problems, achieving significant improvements to the exponent of the ε-dependency in their running times, from roughly d to $d/2$ for the first two problems and from roughly $d/3$ to $d/4$ for problem (iii).

These results are all based on an efficient decomposition of a convex body using a hierarchy of Macbeath regions, and contrast to previous solutions that decompose space using quadtrees and grids. By further application of these techniques, we also show that it is possible to obtain near-optimal preprocessing time for the most efficient data structures to approximately answer queries for (iv) nearest-neighbor searching, (v) directional width, and (vi) polytope membership.

1 Introduction

In this paper we present new faster algorithms to several fundamental geometric approximation problems involving point sets in d-dimensional space. In particular, we present approximation algorithms for ε-kernels, diameter, and bichromatic closest pair. Our results arise from a recently developed shape-sensitive approach to approximating convex bodies, which is based on the classical concept of Macbeath regions. This approach has been applied to computing area-sensitive bounds for polytope approximation [6], polytope approximations with low combinatorial complexity [7], answering approximate polytope-membership queries [8], and approximate nearest-neighbor searching [8]. The results of [8] demonstrated the existence of data structures for these query problems but did not discuss preprocessing in detail. We complete the story by presenting efficient algorithms for building data structures for three related queries: approximate polytope membership, approximate directional width, and approximate nearest-neighbors.

Throughout, we assume that the dimension d is a constant. Our running times will often involve expressions of the form $1/\varepsilon^{\alpha}$. In such cases, $\alpha > 0$ is constant that can be made arbitrarily small. The approximation parameter ε is treated as an asymptotic variable that approaches 0.

We have learned recently of independent results by Timothy Chan for many of the above problems in which the complexity bounds are very similar to ours [17]. Remarkably, the computational techniques seem to be very different, based on Chebyshev polynomials.

The results presented here are based on the upcoming paper on the 33rd International Symposium on Computational Geometry (SoCG 2017).

2 Static Results

Kernel. Given a set S of n points in $\mathbb{R}^d$ and an approximation parameter $\varepsilon > 0$, an *ε-coreset* is an (ideally small) subset of S that approximates some measure over S (see [2] for a survey). Given a nonzero vector $v \in \mathbb{R}^d$, the *directional width* of S in direction v, $\text{width}_v(S)$ is the minimum distance between two hyperplanes that enclose S and are orthogonal to v. A *coreset for the directional width* (also known as an *ε-kernel* and as a *coreset for the extent measure*) is a subset $Q \subseteq S$ such that $\text{width}_v(Q) \geq (1-\varepsilon)\,\text{width}_v(S)$, for all $v \in \mathbb{R}^d$. Kernels are among the most fundamental constructions in geometric approximation, playing a role similar to that of convex hulls in exact computations. Kernels have been used to obtain approximation algorithms to several problems such as diameter, minimum width, convex hull volume, minimum enclosing cylinder, minimum enclosing annulus, and minimum-width cylindrical shell [1, 2].

The concept of ε-kernels was introduced by Agarwal et al. [1]. The existence of ε-kernels with $O(1/\varepsilon^{(d-1)/2})$ points is implied in the works of Dudley [18] and Bronshteyn and Ivanov [15], and this is known to be optimal in the worst case. Agarwal et al. [1] demonstrated how to compute such a kernel in $O(n + 1/\varepsilon^{3(d-1)/2})$ time, which reduces to $O(n)$ when $n = \Omega(1/\varepsilon^{3(d-1)/2})$. While less succinct ε-kernels with

*Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong, arya@cse.ust.hk. Research supported by the Research Grants Council of Hong Kong, China under project number 610012.

†Université d'Auvergne and LIMOS, Clermont-Ferrand, France, fonseca@isima.fr.

‡Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, mount@cs.umd.edu. Research supported by NSF grant CCF–1618866.

$O(1/\varepsilon^{d-1})$ points can be constructed in time $O(n)$ time for all n [1, 13], no linear-time algorithm is known to build an ε-kernel of optimal size. Hereafter, we use the term ε-kernel to refer exclusively to an ε-kernel of size $O(1/\varepsilon^{(d-1)/2})$.

Chan [16] showed that an ε-kernel can be constructed in $O((n+1/\varepsilon^{d-2})\log\frac{1}{\varepsilon})$ time, which is nearly linear when $n = \Omega(1/\varepsilon^{d-2})$. He posed the open problem of obtaining a faster algorithm. A decade later, Arya and Chan [4] showed how to build an ε-kernel in roughly $O(n + \sqrt{n}/\varepsilon^{d/2})$ time using discrete Voronoi diagrams. In this paper, we attain the following near-optimal construction time.

Theorem 1 *Give n points in $\mathbb{R}^d$ and an approximation parameter $\varepsilon > 0$, it is possible to construct an ε-kernel of S with $O(1/\varepsilon^{(d-1)/2})$ points in $O(n\log\frac{1}{\varepsilon} + 1/\varepsilon^{(d-1)/2+\alpha})$ time.*

We note that when $n = o(1/\varepsilon^{(d-1)/2})$, the input S is already an ε-kernel and therefore an $O(n)$ time algorithm is trivial. Because the worst-case output size is $O(1/\varepsilon^{(d-1)/2})$, we may assume that n is at least this large, for otherwise we can simply take S itself to be the kernel. Since $1/\varepsilon^{\alpha}$ dominates $\log\frac{1}{\varepsilon}$, the above running time can be expressed as $O(n/\varepsilon^{\alpha})$, which is nearly linear given that α is arbitrarily small.

Diameter. An important application of ε-kernels is to approximate the diameter of a point set. Given n data points, the *diameter* is defined to be the maximum distance between any two data points. An *ε-approximation* of the diameter is a pair of points whose distance is at least $(1-\varepsilon)$ times the exact diameter. There are multiple algorithms to approximate the diameter [1, 3, 4, 12, 16]. The fastest running times are $O((n+1/\varepsilon^{d-2})\log\frac{1}{\varepsilon})$ [16] and roughly $O(n + \sqrt{n}/\varepsilon^{d/2})$ [4]. The algorithm from [16] essentially computes an ε-kernel Q and then determines the maximum value of $\text{width}_v(Q)$ among a set of $k = O(1/\varepsilon^{(d-1)/2})$ directions v by brute force [1]. Discrete Voronoi diagrams [4] permit this computation in roughly $O(n + \sqrt{n}/\varepsilon^{d/2})$ time. Therefore, combining the kernel construction of Theorem 1 with discrete Voronoi diagrams [4], we reduce n to $O(1/\varepsilon^{(d-1)/2})$ and obtain an algorithm to ε-approximate the diameter in roughly $O(n+1/\varepsilon^{3d/4})$ time. However, we show that it is possible to obtain a much faster algorithm, as presented in the following theorem.

Theorem 2 *Given n points in $\mathbb{R}^d$ and an approximation parameter $\varepsilon > 0$, it is possible to compute an ε-approximation to the diameter of S in $O(n\log\frac{1}{\varepsilon} + 1/\varepsilon^{(d-1)/2+\alpha})$ time.*

Bichromatic Closest Pair. In the *bichromatic closest pair* (BCP) problem, we are given n points from two sets, designated red and blue, and we want to find the closest red-blue pair. In the *ε-approximate* version, the goal is to find a red-blue pair of points whose distance is at most $(1+\varepsilon)$ times the exact BCP distance. Approximations to the BCP problem were introduced in [19], and the most efficient randomized approximation algorithm runs in roughly $O(n/\varepsilon^{d/3})$ expected time [4]. We present the following result.

Theorem 3 *Given n red and blue points in $\mathbb{R}^d$ and an approximation parameter $\varepsilon > 0$, there is a randomized algorithm that computes an ε-approximation to the BCP in $O(n/\varepsilon^{d/4+\alpha})$ expected time.*

3 Data Structure Results

Polytope membership. Let P denote a convex polytope in $\mathbb{R}^d$, represented as the bounded intersection of n halfspaces. The *polytope membership problem* consists of preprocessing P so that it is possible to determine efficiently whether a given query point $q \in \mathbb{R}^d$ lies within P. In the ε-approximate version, we consider an expanded convex body $K \supset P$. A natural way to define this expansion would be to consider the set of points that lie within distance $\varepsilon \cdot \text{diam}(P)$ of P, thus defining a body whose Hausdorff distance from P is $\varepsilon \cdot \text{diam}(P)$. However, this definition has the shortcoming that it is not sensitive to the directional width of P. Instead, we define K as follows. For any nonzero vector $v \in \mathbb{R}^d$, consider the two supporting hyperplanes for P that are normal to v. Translate each of these hyperplanes outward by a distance of $\varepsilon \cdot \text{width}_v(P)$, and consider the closed slab-like region lying between them. Define K to be the intersection of this (infinite) set of slabs. This is clearly a stronger approximation than the Hausdorff-based definition. An *ε-approximate polytope membership query* (ε-APM query) returns a positive result if the query point q is inside P, a negative result if q is outside K, and may return either result otherwise.[1]

We recently proposed an optimal data structure to answer approximate polytope membership queries, but efficient preprocessing remained an open problem [8]. In this paper, we present a similar data structure that not only attains optimal storage and query time, but can also be preprocessed in near-optimal time.

Theorem 4 *Given a convex polytope P in $\mathbb{R}^d$ represented as the intersection of n halfspaces and an approximation parameter $\varepsilon > 0$, there is a data structure that can answer ε-APM queries with query time $O(\log\frac{1}{\varepsilon})$, space $O(1/\varepsilon^{(d-1)/2})$, and preprocessing time $O(n\log\frac{1}{\varepsilon} + 1/\varepsilon^{(d-1)/2+\alpha})$.*

[1] Our earlier works on ε-APM queries [5, 8] use the weaker Hausdorff form to define the problem, but the solutions presented there actually achieve the stronger direction-sensitive form.

Directional width. Applying the previous data structure in the dual space, we obtain a data structure for the following *ε-approximate directional width* problem, which is closely related to ε-kernels. Given a set S of n points in $\mathbb{R}^d$ and an approximation parameter $\varepsilon > 0$, the goal is to preprocess S to efficiently ε-approximate $\text{width}_v(S)$, for a nonzero query vector v. We present the following result.

Theorem 5 *Given n points in $\mathbb{R}^d$ and an approximation parameter $\varepsilon > 0$, there is a data structure that can answer ε-approximate directional width queries with query time $O(\log^2 \frac{1}{\varepsilon})$, space $O(1/\varepsilon^{(d-1)/2})$, and preprocessing time $O(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{(d-1)/2+\alpha})$.*

Nearest Neighbor. Let S be a set of n points in $\mathbb{R}^d$. Given any $q \in \mathbb{R}^d$, an *ε-approximate nearest neighbor* (ANN) of q is any point of S whose distance from q is at most $(1+\varepsilon)$ times the distance to q's closest point in S. The objective is to preprocess S in order to answer such queries efficiently. Data structures for approximate nearest neighbor searching (in fixed dimensions) have been proposed by several authors, offering space-time tradeoffs (see [8] for an overview of the tradeoffs). Applying the reduction from approximate nearest neighbor to approximate polytope membership established in [5] together with Theorem 4, we obtain the following result, which matches the best bound [8] up to an $O(\log \frac{1}{\varepsilon})$ factor in the query time, but offers faster preprocessing time.

Theorem 6 *Given n points in $\mathbb{R}^d$, an approximation parameter $\varepsilon > 0$, and m such that $\log \frac{1}{\varepsilon} \leq m \leq 1/(\varepsilon^{d/2} \log \frac{1}{\varepsilon})$, there is a data structure that can answer ε-ANN queries with query time $O(\log n + (\log \frac{1}{\varepsilon})/(m \cdot \varepsilon^{d/2}))$ space $O(n\,m)$, and preprocessing time $O(n \log n \log \frac{1}{\varepsilon} + n\,m/\varepsilon^{\alpha})$.*

4 Techniques

In contrast to previous kernel constructions, which are based on grids and the execution of Bronshteyn and Ivanov's algorithm, our construction employs a classical structure from the theory of convexity, called *Macbeath regions* [20]. Macbeath regions have found numerous uses in the theory of convex sets and the geometry of numbers (see Bárány [11] for an excellent survey). They have also been applied to several problems in the field of computational geometry. However, most previous results were either in the form of lower bounds [9, 10, 14] or focused on existential results [6, 7, 21].

In [8] the authors introduced a data structure based on a hierarchy of ellipsoids based on Macbeath regions to answer approximate polytope membership queries, but the efficient computation of the hierarchy was not considered. In this paper, we show how to efficiently construct the Macbeath regions that form the basis of this hierarchy.

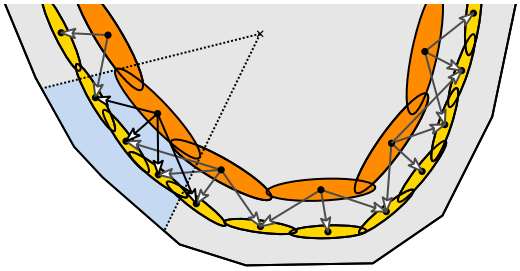


Figure 1: Two levels of the hierarchy of ellipsoids based on Macbeath regions.

Let P denote a convex polytope in $\mathbb{R}^d$. Each level i in the hierarchy corresponds to a δ_i-approximation of the boundary of P by a set of $O(1/\delta_i^{(d-1)/2})$ ellipsoids, where $\delta_i = \Theta(1/2^i)$. Each ellipsoid has $O(1)$ children, which correspond to the ellipsoids of the following level that approximate the same portion of the boundary (see Figure 1). The hierarchy starts with $\delta_0 = \Theta(1)$ and stops after $O(\log \frac{1}{\delta})$ levels when $\delta_i = \delta$, for a desired approximation δ. We present a simple algorithm to construct the hierarchy in $O(n + 1/\delta^{3(d-1)/2})$ time. The polytope P can be presented as either the intersection of n halfspaces or the convex hull of n points.

Our algorithm to compute an ε-kernel in time $O(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{(d-1)/2+\alpha})$ (Theorem 1) is conceptually quite simple. Since the time to build the ε-approximation hierarchy for the convex hull is prohibitively high, we use an approximation parameter $\delta = \varepsilon^{1/3}$ to build a δ-approximation hierarchy in $O(n + 1/\varepsilon^{(d-1)/2})$ time. By navigating through this hierarchy, we partition the n points among the leaf Macbeath ellipsoids in $O(n \log \frac{1}{\varepsilon})$ time, discarding points that are too far from the boundary. We then compute an (ε/δ)-kernel for the set of points in each leaf ellipsoid and return the union of the kernels computed.

Given an algorithm to compute an ε-kernel in $O(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{t(d-1)})$ time, the previous procedure produces an ε-kernel in $O(n \log \frac{1}{\varepsilon} + 1/\varepsilon^{t'(d-1)})$ time where $t' = (4t+1)/6$. Bootstrapping the construction a constant number of times, the value of t goes down from 1 to a value that is arbitrarily close to $1/2$. This discrepancy accounts for the $O(1/\varepsilon^{\alpha})$ factors in our running times.

To prove Theorem 4, we use our kernel construction in the dual space to efficiently build a polytope membership data structure. The key idea is to compute multiple kernels in order to avoid examining the whole polytope. We build a δ-approximate hierarchy (for a proper value of δ) in $O(n + 1/\delta^{3(d-1)/2})$ time. Each leaf node of the data structure is associated with a certain portion of the polytope, called a *shadow*.

We then build an (ε/δ)-kernel (in the dual space) for the shadow of each leaf node, followed by an (ε/δ)-approximate polytope membership data structure for each kernel. Given a query point q, the δ-approximate hierarchy is able to either correctly answer the query (to the desired approximation $\varepsilon \leq \delta$) or to locate a leaf shadow that contains the query point. In the latter case, we transfer the query to the data structure associated with that leaf node. The aforementioned construction reduces the preprocessing time for an approximate polytope membership data structure. Again, we use bootstrapping to obtain a near-optimal preprocessing time.

The remaining theorems follow from Theorems 1 and 4, together with several known reductions.

References

[1] P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan. Approximating extent measures of points. *J. Assoc. Comput. Mach.*, 51:606–635, 2004.

[2] P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan. Geometric approximation via coresets. In J. E. Goodman, J. Pach, and E. Welzl, editors, *Combinatorial and Computational Geometry*. MSRI Publications, 2005.

[3] P. K. Agarwal, J. Matoušek, and S. Suri. Farthest neighbors, maximum spanning trees and related problems in higher dimensions. *Computational Geometry*, 1(4):189–201, 1992.

[4] S. Arya and T. M. Chan. Better ε-dependencies for offline approximate nearest neighbor search, Euclidean minimum spanning trees, and ε-kernels. In *Proc. 30th Annu. Sympos. Comput. Geom.*, pages 416–425, 2014.

[5] S. Arya, G. D. da Fonseca, and D. M. Mount. Approximate polytope membership queries. In *Proc. 43rd Annu. ACM Sympos. Theory Comput.*, pages 579–586, 2011. (Full version available from `http://arxiv.org/abs/1604.01183`, to appear on SIAM J. Computing).

[6] S. Arya, G. D. da Fonseca, and D. M. Mount. Optimal area-sensitive bounds for polytope approximation. In *Proc. 28th Annu. Sympos. Comput. Geom.*, pages 363–372, 2012.

[7] S. Arya, G. D. da Fonseca, and D. M. Mount. On the combinatorial complexity of approximating polytopes. In *Proc. 32nd Internat. Sympos. Comput. Geom.*, pages 11:1–11:15, 2016. (Full version `http://arxiv.org/abs/1604.01175`, to appear on Discrete Comput. Geom.).

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

[9] S. Arya, T. Malamatos, and D. M. Mount. The effect of corners on the complexity of approximate range searching. *Discrete Comput. Geom.*, 41:398–443, 2009.

[10] S. Arya, D. M. Mount, and J. Xia. Tight lower bounds for halfspace range searching. *Discrete Comput. Geom.*, 47:711–730, 2012.

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

[12] G. Barequet and S. Har-Peled. Efficiently approximating the minimum-volume bounding box of a point set in three dimensions. *Journal of Algorithms*, 38(1):91–109, 2001.

[13] J. L. Bentley, M. G. Faust, and F. P. Preparata. Approximation algorithms for convex hulls. *Commun. ACM*, 25(1):64–68, 1982.

[14] H. Brönnimann, B. Chazelle, and J. Pach. How hard is halfspace range searching. *Discrete Comput. Geom.*, 10:143–155, 1993.

[15] E. M. Bronshteyn and L. D. Ivanov. The approximation of convex sets by polyhedra. *Siberian Math. J.*, 16:852–853, 1976.

[16] T. M. Chan. Faster core-set constructions and data-stream algorithms in fixed dimensions. *Comput. Geom. Theory Appl.*, 35(1):20–35, 2006.

[17] T. M. Chan. Applications of Chebyshev polynomials to low-dimensional computational geometry. In *Proc. 33rd Internat. Sympos. Comput. Geom.*, 2017.

[18] R. M. Dudley. Metric entropy of some classes of sets with differentiable boundaries. *J. Approx. Theory*, 10(3):227–236, 1974.

[19] S. Khuller and Y. Matias. A simple randomized sieve algorithm for the closest-pair problem. *Information and Computation*, 118(1):34–37, 1995.

[20] A. M. Macbeath. A theorem on non-homogeneous lattices. *Ann. of Math.*, 56:269–293, 1952.

[21] N. H. Mustafa and S. Ray. Near-optimal generalisations of a theorem of Macbeath. In *Proc. 31st Internat. Sympos. on Theoret. Aspects of Comp. Sci.*, pages 578–589, 2014.

A Generic Method for Finding Coresets for Clustering Problems*

Mikkel Abrahamsen†, Mark de Berg‡, Kevin Buchin‡, Mehran Mehr‡, Ali D. Mehrabi‡

Abstract

In a geometric k-clustering problem the goal is to partition a set of n points S in $\mathbb{R}^d$ into k subsets such that a certain cost function of the clustering is minimized. We present a general method to compute a small coreset R of size independent of n such that the cost of a k-clustering of R is a $(1+\varepsilon)$-approximation to the cost of k-clustering of S. Our method applies to a large class of clustering problems, including k-center clustering in any L_p-metric. Our algorithm has a linear running time for a fixed k.

1 Introduction

Clustering is a fundamental task in data analysis. It involves partitioning a given data set into subsets called *clusters*, such that similar elements end up in the same cluster. Often the data elements can be viewed as points in a geometric space, and similarity is measured by considering the distance between the points. We focus on clustering problems of the following type. Let S be a set of n points in $\mathbb{R}^d$, and let $k \geqslant 2$ be a natural number. A *k-clustering* of S is a partitioning $\mathcal{C}$ of S into at most k clusters. Let $\Phi(\mathcal{C})$ denote the *cost* of $\mathcal{C}$. The goal is now to find a clustering $\mathcal{C}$ that minimizes $\Phi(\mathcal{C})$. Many well-known geometric clustering problems are of this type. Among them is the k-center problem. In the *Euclidean k-center problem*, $\Phi(\mathcal{C})$ is the maximum cost of any of the clusters $C \in \mathcal{C}$, where the cost of C is the radius of its smallest enclosing ball. Hence, in the Euclidean k-center problem we want to cover the point set S by k congruent balls of minimum radius. The *rectilinear k-center problem* is defined similarly except that one considers the L_∞-metric; thus we want to cover S by k congruent axis-aligned cubes[1] of minimum size.

*MA is partly supported by Mikkel Thorup's Advanced Grant from the Danish Council for Independent Research under the Sapere Aude research career programme. MdB, KB, MM, and AM are supported by the Netherlands' Organisation for Scientific Research (NWO) under project no. 024.002.003, 612.001.207, 022.005025, and 612.001.118 respectively.

†University of Copenhagen, mia@di.ku.dk.

‡TU Eindhoven, mdberg@win.tue.nl, k.a.buchin@tue.nl, m.mehr@tue.nl, amehrabi@win.tue.nl.

[1]Throughout the paper, when we speak of cubes (or squares, or rectangles, or boxes) we always mean axis-aligned cubes (or squares, or rectangles, or boxes).

Our contribution. Our main result is a general method to find an ε-coreset for a given clustering problem. Given a set of points S, a cost function Φ, the number of clusters k, a value $\varepsilon > 0$, an ε-coreset is a set $R \subseteq S$ such that $\Phi(\mathcal{C}'_{\text{opt}}) \leqslant (1+\varepsilon) \cdot \Phi(\mathcal{C}_{\text{opt}})$, where $\mathcal{C}'_{\text{opt}}$ is an optimal clustering for R and $\mathcal{C}_{\text{opt}}$ is an optimal clustering for S.

In [1], we present a data structure for storing a set S of n points in $\mathbb{R}^d$ that can answer approximate range-clustering queries: given a rectangular query range Q, the data structure can report a $(1+\varepsilon)$-approximate solution to the k-clustering problem of the point set $S \cap Q$. That algorithm is based on finding an ε-coreset for k-clustering on $S \cap Q$; assuming fixed k and ε, the query time for finding an ε-coreset of size $O(k\,(f(k)/\varepsilon)^d)$ for $S \cap Q$ is $O(\log^{d-1} n)$ and the preprocessing time is $O(n \log^{d-1} n)$, where $f(k)$ is a function that only depends on the number of clusters.

Here we consider the off-line version of the problem instead of the data-structuring version. We find a coreset of the same size for a set of points S. The algorithm runs in linear time for constant k, assuming a certain model of computing. Our algorithm borrows some of the ideas from [1]. In particular, Lemmas 2 and 3 are from [1]. This is similar to the approach taken by Har-Peled and Mazumdar [3], who solve the approximate k-means and k-median problem efficiently by generating a coreset of size $O((k/\varepsilon^d) \cdot \log n)$.

Similar to [1], our method applies to a large class of clustering problems including the k-center problem in any L_p-metric, variants of the k-center problem where we want to minimize the sum (rather than maximum) of the cluster radii, and the 2-dimensional problem where we want to minimize the maximum or sum of the perimeters of the clusters.

2 The algorithm

We start by defining the class of clustering problems to which our algorithm applies.

Let S be a set of n points in $\mathbb{R}^d$ and let $\text{Part}(S)$ be the set of all partitions of S. Let $\text{Part}_k(S)$ be the set of all partitions into at most k subsets, that is, all k-clusterings of S. Let $\Phi : \text{Part}(S) \mapsto \mathbb{R}_{\geqslant 0}$ be the cost function defining our clustering problem, and define

$$\textsc{Opt}_k(S) := \min_{\mathcal{C} \in \text{Part}_k(S)} \Phi(\mathcal{C})$$

to be the minimum cost of any k-clustering. An ε-coreset for this clustering problem defined by Φ is

a set $R \subseteq S$ such that $|R|$ is independent of n and $\textsc{Opt}_k(R) \leqslant (1+\varepsilon) \cdot \textsc{Opt}_k(S)$.

To define the class of clusterings to which our method applies, we need the concept of r-packing [2]. Actually, we use a slightly weaker variant, which we define as follows. Let $|pq|$ denote the Euclidean distance between two points p and q. A subset $R \subseteq P$ of a point set P is called a *weak r-packing* for P, for some $r > 0$, if for any point $p \in P$ there exists a *packing point* $q \in R$ such that $|pq| \leqslant r$. (The difference with standard r-packings is that we do not require that $|qq'| > r$ for any two points $q, q' \in R$.) The clustering problems to which our method applies are the ones whose cost function is *regular*, as defined next.

Definition 1 *A cost function $\Phi : \text{Part}(S) \mapsto \mathbb{R}_{\geqslant 0}$ is called* $(c, f(k))$-regular*, if there is constant c and function $f : \mathbb{N}_{\geqslant 2} \mapsto \mathbb{R}_{\geqslant 0}$ such that the following holds.*

- *For any clustering $\mathcal{C} \in \text{Part}(S)$, we have*

$$\Phi(\mathcal{C}) \geqslant c \cdot \max_{C \in \mathcal{C}} \text{diam}(C),$$

  where $\text{diam}(C) = \max_{p,q \in C} |pq|$ denotes the Euclidean diameter of the cluster C. We call this the diameter-sensitivity *property.*

- *For any weak r-packing R of S, and any $k \geqslant 2$, we have that*

$$\textsc{Opt}_k(R) \leqslant \textsc{Opt}_k(S) \leqslant \textsc{Opt}_k(R) + r \cdot f(k).$$

Example. We show k-center problems have regular cost functions. For a cluster C, let $\text{radius}_p(C)$ denote the radius of the minimum enclosing ball of C in the L_p-metric. In the L_∞ metric, for instance, $\text{radius}_p(C)$ is half the edge length of a minimum enclosing axis-aligned cube of C. Then the cost of a clustering $\mathcal{C}$ for the k-center problem in the L_p-metric is $\Phi_p^{\max}(\mathcal{C}) = \max_{C \in \mathcal{C}} \text{radius}_p(C)$. One easily verifies that the cost function for the rectilinear k-center problem is $(1/(2\sqrt{d}), 1)$-regular, and for the Euclidean k-center problem it is $(1/2, 1)$-regular. (In fact $\Phi_p^{\max}(\mathcal{C})$ is regular for any p.)

Overview of the algorithm and the main lemmas. We start with a high-level overview of our approach. Let S be the given point set for which we want to find a coreset. Our algorithm is as follows. Note

Algorithm 1 $\textsc{FindCoreset}(S, k, \varepsilon)$

1: Compute a lower bound LB on $\textsc{Opt}_k(S)$.
2: Set $r := \varepsilon \cdot \text{LB}/f(k)$
3: Compute a weak r-packing R on S.
4: **return** R

that R is the desired ε-coreset because Φ is $(c, f(k))$-regular. The following lemma is immediate.

Lemma 1 $\textsc{Opt}_k(R) \leqslant (1+\varepsilon) \cdot \textsc{Opt}_k(S)$.

Next, we will show how to perform Steps 1 and 3: we will describe an algorithm that allows us to compute a suitable lower bound LB and a corresponding weak r-packing, such that the size of the r-packing depends only on ε and k but not on $|S|$.

Our lower bound and weak packing computation are based on so-called cube covers. A *cube cover* of S is a collection $\mathcal{B}$ of interior-disjoint cubes that together cover all the points in S and such that each $B \in \mathcal{B}$ contains at least one point from S (in its interior or on its boundary). Define the size of a cube B, denoted by $\text{size}(B)$, to be its edge length. The following lemma follows immediately from the fact that the diameter of a cube B in $\mathbb{R}^d$ is $\sqrt{d} \cdot \text{size}(B)$.

Lemma 2 (Abrahamsen *et al.* [1]) *Let $\mathcal{B}$ be a cube cover of S such that $\text{size}(B) \leqslant r/\sqrt{d}$ for all $B \in \mathcal{B}$. Then any subset $R \subseteq S$ containing a point from each cube $B \in \mathcal{B}$ is a weak r-packing for S.*

Our next lemma shows we can find a lower bound on $\textsc{Opt}_k(S)$ from a suitable cube cover.

Lemma 3 (Abrahamsen *et al.* [1]) *Suppose the cost function Φ is $(c, f(k))$-regular. Let $\mathcal{B}$ be a cube cover of S such that $|\mathcal{B}| > k2^d$. Then $\textsc{Opt}_k(S) \geqslant c \cdot \min_{B \in \mathcal{B}} \text{size}(B)$.*

Proof. For two cubes B, B' such that the maximum x_i-coordinate of B is at most the minimum x_i-coordinate of B', we say that B is *i-below* B' and B' is *i-above* B. We denote this relation by $B \prec_i B'$. Now consider an optimal k-clustering $\mathcal{C}_{\text{opt}}$ of S. By the pigeonhole principle, there is a cluster $C \in \mathcal{C}_{\text{opt}}$ containing points from at least $2^d + 1$ cubes. Let $\mathcal{B}_C$ be the set of cubes that contain at least one point in C.

Clearly, if there are cubes $B, B', B'' \in \mathcal{B}_C$ such that $B' \prec_i B \prec_i B''$ for some $1 \leqslant i \leqslant d$, then the cluster C contains two points at distance at least $\text{size}(B)$ from each other. Since Φ is $(c, f(k))$-regular this implies that $\Phi(\mathcal{C}_{\text{opt}}) \geqslant c \cdot \text{size}(B)$, which proves the lemma.

Now suppose for a contradiction that such a triple B', B, B'' does not exist. Then we can define a characteristic vector $\Gamma(B) = (\Gamma_1, \ldots, \Gamma_d(B))$ for each cube $B \in \mathcal{B}_C$, as follows:

$$\Gamma_i(B) = \begin{cases} 1 & \text{if no cube } B' \in \mathcal{B}_C \text{ is } i\text{-below } B \\ 0 & \text{if no cube } B'' \in \mathcal{B}_C \text{ is } i\text{-above } B \end{cases}$$

Since the number of distinct characteristic vectors is $2^d < |B_C|$, there must be two cubes $B_1, B_2 \in B_C$ with identical characteristic vectors. However, any two interior-disjoint cubes can be separated by an axis-parallel hyperplane, so there is at least one $i \in \{1, \ldots, d\}$ such that B_1 is i-below or i-above B_2. But this contradicts that $\Gamma(B_1) = \Gamma(B_2)$. □

Details of Steps 1 and 3. For simplicity, we continue the discussion for 2-dimensional case. Generalizing the results to higher dimension is not hard.

Let $G(\alpha)$ denote the *grid* in $\mathbb{R}^2$ whose cells have size α and for which the origin O is a grid point. We call α, the *width* of grid $G(\alpha)$. We define cells to be open on the right and top, and closed on the left and bottom, so a cell is of the form $[\alpha i, \alpha(i+1)) \times [\alpha j, \alpha(j+1))$ for integers i, j. Notice that Lemmas 2 and 3 are still valid for such partially open cells instead of closed cells.

For any value α, *id* of a point $p = (x, y)$ is defined as $\mathrm{id}(p, \alpha) = (\lfloor x/\alpha \rfloor, \lfloor y/\alpha \rfloor)$. Clearly, only points in the same cell of $G(\alpha)$, have similar id values. Using their id, we can store and fetch a set of points inside a grid efficiently, by using hashing.

For an integer s, we define a *canonical grid* with parameter s to be $G(2^s)$ and denote it by G_s. Similarly, for a point $p \in \mathbb{R}^2$ we define $\mathrm{id}_s(p) := \mathrm{id}(p, 2^s)$. A *canonical square* is any cell of one of the grids G_s. W.l.o.g, we assume all the points in a given set of points S reside in a single canonical square of some canonical grid.

We define the *separation level* of two points $p_1, p_2 \in \mathbb{R}^2$ to be the smallest integer s, such that p_1 and p_2 reside in a single canonical square in G_s and denote it by $sdist(p_1, p_2)$. We can compute this number in constant time in a computational model where computing $\lg x$,2^x and $\lfloor x \rfloor$ takes constant time ($\lg x = \log_2 x$). See Section 2.2.2 of [2] for more details.

We define the *canonical closest pair* of a set of points S to be the pair of points $p_1, p_2 \in S$ with minimum separation level. We define the *canonical closest pair distance* of S to be $sdist(p_1, p_2)$ or equivalently the minimum integer s such that at least two points in S are in the same cell of G_s.

Throughout this paper, we use a data-structure H that stores a set of points indexed on their id in an arbitrary grid G by using hashing. H can store just one point for each cell of G. We denote the number of points in H by $|H|$. These are the operations supported by H (all times are expected):

1. Get the size, $|H|$, in $O(1)$ time.
2. Test if a cell of G is empty in $O(1)$ time.
3. Add a point to an empty cell of G in $O(1)$ time.
4. Retrieve all the points in H in $O(|H|)$ time.

We can also 'Rebuild H for a new grid G' by retrieving all points from the current data structure H, and inserting them one by one into a new structure H based on the grid G. If we wish to insert a point p but we already inserted another point with the same id, then p is discarded.

Computing canonical closest pair distance. Our algorithm is similar to the algorithm for computing *Euclidean closest pair* distance described in Section 1.2 of [2].

Algorithm 2 CCLOSESTPAIR(S)

```
1: Compute a random permutation p_1, ..., p_n of S.
2: Set d_0 := ∞, s := ∞, and H to empty
3: for i := 1 to n do
4:     if H[id_s(p_i)] is empty then
5:         Set d_i := d_{i-1}
6:         Set H[id_s(p_i)] := p_i
7:     else
8:         Set d_i := sdist(p_i, H[id_s(p_i)])
9:         Set s := d_i − 1, and rebuild H for G_s
10:    end if
11: end for
12: return d_n
```

Let d_i be the canonical closest pair distance of $p_1, \ldots, p_i$. To compute d_i from d_{i-1}, consider the points $p_1, \ldots, p_{i-1}$ in the grid G_s where $s = d_{i-1} - 1$. Obviously, no two points are in the same cell by definition of closest canonical pair distance. Now, if the point p_i is the only point in its cell (of G_s), it means the separation level of p_i to any other point in $p_1, \ldots, p_{i-1}$ is at least d_{i-1} and therefore $d_i = d_{i-1}$. Otherwise, if p_i and p_j, where $1 \leqslant j < i$, are in the same cell of G_s, then $d_i = sdist(p_i, p_j)$.

Lemma 4 *Algorithm* CCLOSESTPAIR *computes the canonical closest pair distance of a set of n points S in expected $O(n)$ time.*

Proof. As discussed above, after execution of the i-th iteration, the canonical closest pair distance of $p_1, \ldots, p_i$ is stored in the variable d_i. This proves the correctness of the algorithm.

For each point p_i, we only need to rebuild H in time $O(i)$ if $d_i < d_{i-1}$; otherwise we spend $O(1)$ time for handling it. As the points are in random order, we have $\Pr[d_i < d_{i-1}] \leqslant 2/i$ for $i \geqslant 3$; therefore, the expected running time is

$$O(n) + \sum_{i=3}^{n} \Pr[d_i < d_{i-1}] \cdot O(i) = O(n)$$

□

Step 1 of FINDCORESET. We start with the grid G_s where $s = -\infty$ and add points in S one by one to this grid. Whenever the number of non-empty cells exceeds $16k$, we increase the parameter s of the grid G_s so that the number of non-empty cells of the new grid G_s is still greater that $4k$ but not greater than $16k$. After adding all the points S to grid G_s we can use Lemma 3 to lower bound $\mathrm{OPT}_k(S)$.

Algorithm 3 LowerBound(S, k)

```
1: Compute a random permutation p_1, ..., p_n of S.
2: Set s := -∞, and H to empty
3: for i := 1 to n do
4:     if H[id_s(p_i)] is empty then
5:         Set H[id_s(p_i)] := p_i
6:         if |H| > 16k then
7:             Set s := CClosestPair(H)
8:             Rebuild H for the new grid G_s
9:         end if
10:    end if
11: end for
12: return c · 2^s
```

Lemma 5 *Given a set of points S and a value $k \geqslant 2$, the algorithm* LowerBound *computes a lower bound for $\text{Opt}_k(S)$ in expected $O(n + k^2 \log(n/k))$ time. Moreover, after execution of this algorithm G_s contains at most $16k$ nonempty cells.*

Proof. After execution of the algorithm, if $s = -\infty$ the algorithm returns 0 which is an obvious lower bound. Otherwise we show that $|H| > 4k$.

If $s \neq -\infty$, it means the value of s has changed in line 7. Whenever the value of s changes in line 7, each cell of G_s contains at most 4 points of H, because the new value of s is canonical closest pair distance of $p_1, \ldots, p_i$. Therefore, after rebuilding H in line 8 the number of points in H decreases by a factor of at most 4. Since before a rebuild operation, we have $|H| > 16k$, after rebuild we have $|H| > 4k$ and Lemma 3 and Definition 1 proves the correctness of the lower bound returned by the algorithm.

Notice that $|H| > 16k$ only after adding a point to an empty cell of G_s and even at that time $|H| = 16k+1$ and after execution of lines 7 and 8, we have $|H| \leqslant 16k$, because in the new grid G_s at least one of the canonical closest pair will be in the same cell and thus will be ignored by H. Therefore, the claim that G_s contains at most $16k$ nonempty cells is correct.

To compute the running time of the algorithm, notice that the most time consuming operations of the algorithm are lines 7 and 8, both of them run in $O(k)$ time. For each point p_i the lines 7 and 8 can get executed only if p_i is alone in G_s, i.e. p_i is the only point in its cell (of G_s) among the points $p_1, \ldots, p_i$; otherwise we handle p_i in constant time. We know that after adding p_i to H, we have $|H| \leqslant 16k+1$; therefore among $p_1, \ldots, p_i$ there can be at most $16k + 1$ points that are alone in their cell in G_s. As the points are in random order, $\Pr[p_i \text{ is alone}] \leqslant (16k + 1)/i$, for $i > 16k$. Hence, the expected running time is

$$O(n) + \sum_{i=16k}^{n} \Pr[p_i \text{ is alone}] \cdot O(k) = O\left(n + k^2 \log \frac{n}{k}\right)$$

□

Step 3 of FindCoreset. The following algorithm computes a weak r-packing in linear time by selecting one point from each nonempty cell of the grid $G(r/\sqrt{2})$. The correctness of the algorithm follows from Lemma 2.

Algorithm 4 WeakPacking(S, r)

```
1: Set α := r/√2, and H to empty
2: for i := 1 to n do
3:     if H[id(p_i, α)] is empty then
4:         Set H[id(p_i, α)] := p_i
5:     end if
6: end for
7: return H
```

Putting everything together. The running time of Algorithm FindCoreset is dominated by the running time of Algorithm LowerBound, which is $O(n + k^2 \log(n/k))$. Assuming k is constant, FindCoreset runs in linear time.

By Lemma 5 there exists a cube cover of S using at most $16k$ cells of the grid $G(\text{LB}/c)$. Therefore, the grid $G(r/\sqrt{2})$ where $r := \varepsilon \cdot \text{LB}/f(k)$ has at most $16k \cdot \left(\left(\sqrt{2} f(k)/(c\,\varepsilon)\right) + 1\right)^2$ nonempty cells and Algorithm WeakPacking cannot return more than $O(k\,(f(k)/\varepsilon)^2)$ points in Step 3 of FindCoreset.

As stated earlier, it is easy to generalize the algorithms and lemmas described above to higher dimension. Therefore, we can state the following theorem,

Theorem 6 *Let S be a set of n points in $\mathbb{R}^d$ and let Φ be a $(c, f(k))$-regular cost function. Then, it is possible to find an ε-coreset R for k-clustering according to Φ, of size $O(k\,(f(k)/\varepsilon)^d)$ in expected $O(n + k^2 \log(n/k))$ time, for values $k \geqslant 2$ and $\varepsilon > 0$.*

Corollary 1 *Let S be a set of n points in $\mathbb{R}^d$. It is possible to find an ε-coreset R for Euclidean (rectilinear) k-center, of size $O(k/\varepsilon^d)$ in expected $O(n + k^2 \log(n/k))$ time, for values $k \geqslant 2$ and $\varepsilon > 0$.*

References

[1] M. Abrahamsen, M. de Berg, K. Buchin, M. Mehr, and A. D. Mehrabi. Range-Clustering Queries. *33rd Symp. on Comput. Geom. (SoCG)*, 2017.

[2] S. Har-Peled. *Geometric approximation algorithms*, volume 173 of *Mathematical surveys and monographs*. American Mathematical Society, 2011.

[3] S. Har-Peled and S. Mazumdar. On coresets for k-means and k-median clustering. In *36th Annu. ACM Sympos. Theory Comput. (STOC)*, pages 291–300, 2004.

Range-Clustering Queries*

Mikkel Abrahamsen†, Mark de Berg‡, Kevin Buchin‡, Mehran Mehr‡, Ali D. Mehrabi‡

Abstract

Given a planar point set P and an integer $k \geqslant 2$, the rectilinear k-center problem asks for at most k congruent axis-aligned squares of minimal size whose union covers P. In this short note, we study the data-structuring version of the problem in which one is allowed to preprocess P into a data structure that can answer the following type of queries quickly. Given an axis-aligned query rectangle Q and an integer $k \geqslant 2$, solve the rectilinear k-center for $P \cap \mathsf{Q}$. We present two linear-size data structures for queries in $\mathbb{R}^1$; one has $O(k^2 \log^2 n)$ query time, the other has $O(3^k \log n)$ query time. For queries in $\mathbb{R}^2$, we present a data structure that answers 2-center queries in $O(\log n)$ time, and one that answers 3-center queries in $O(\log^2 n)$ time. Both data structures use $O(n \log^\varepsilon n)$ storage.

1 Introduction

The range-searching problem is one of the most important and widely studied problems in computational geometry. In the standard setting one is given a set P of points, and a query asks to report or count all points inside a geometric query range Q. In many applications, however, one would like to perform further analysis on the set $P \cap \mathsf{Q}$, to obtain more information about its structure. Currently one then has to proceed as follows: first perform a range-reporting query to explicitly report $P \cap \mathsf{Q}$, then apply a suitable analysis algorithm to $P \cap \mathsf{Q}$. This two-stage process can be quite costly, because algorithms for analyzing geometric data sets can be slow and $P \cap \mathsf{Q}$ can be large. To avoid this we would need data structures for what we call *range-analysis queries*, which directly compute the desired structural information about $P \cap \mathsf{Q}$. In this paper we develop such data structures for the case where one is interested in a cluster-analysis of $P \cap \mathsf{Q}$.

Clustering is a fundamental task in data analysis. It involves partitioning a given data set into subsets called clusters, such that similar elements end up in the same cluster. Often the data elements can be viewed as points in a geometric space, and similarity is measured by considering the distance between the points. We focus on clustering problems of the following type. Let P be a set of n points in $\mathbb{R}^d$, and let $k \geqslant 2$ be a natural number. A *k-clustering* of P is a partitioning $\mathcal{C}$ of P into at most k clusters. Let $\Phi(\mathcal{C})$ denote the *cost* of $\mathcal{C}$. The goal is now to find a clustering $\mathcal{C}$ that minimizes $\Phi(\mathcal{C})$. Many well-known geometric clustering problems are of this type. Among them is the k-center problem. In the *Euclidean k-center problem* $\Phi(\mathcal{C})$ is the maximum cost of any of the clusters $C \in \mathcal{C}$, where the cost of C is the radius of its smallest enclosing ball. Hence, in the Euclidean k-center problem we want to cover the point set P by k congruent balls of minimum radius. The *rectilinear k-center problem* is defined similarly except that one considers the ℓ_∞-metric; thus we want to cover P by k congruent axis-aligned cubes[1] of minimum size. The k-center problem, including the important special case of the 2-center problem, has been studied extensively, both for the Euclidean case (e.g. [3]) and for the rectilinear case (e.g. [4, 5]).

Our contribution. Due to space limitation, in this short note we are unable to present most of our results. We instead highlight a global picture of our main technique in obtaining most of our results, and discuss our results for a special case—namely the results mentioned in the abstract—in details.

Our main result is a general method to answer *approximate* orthogonal range-clustering queries in $\mathbb{R}^d$. Here the query specifies (besides the query box Q and the number of clusters k) a value $\varepsilon > 0$; the goal then is to compute a k-clustering $\mathcal{C}$ of $P \cap \mathsf{Q}$ with $\Phi(\mathcal{C}) \leqslant (1+\varepsilon) \cdot \Phi(\mathcal{C}_{\text{opt}})$, where $\mathcal{C}_{\text{opt}}$ is an optimal clustering for $P \cap \mathsf{Q}$. Our method works by computing a sample $R \subseteq P \cap \mathsf{Q}$ such that solving the problem on R gives us the desired approximate solution. We show that for a large class of cost functions Φ we can find such a sample of size only $O(k(f(k)/\varepsilon)^d)$, where $f(k)$ is a function that only depends on the number of clusters. A key step in our method is a procedure to efficiently compute a lower bound on the value of an optimal solution within the query range. The class of clustering problems to which our

*MA is partly supported by Mikkel Thorup's Advanced Grant from the Danish Council for Independent Research under the Sapere Aude research career programme. MdB, KB, MM, and AM are supported by the Netherlands Organization for Scientific Research (NWO) under project no. 024.002.003, 612.001.207, 022.005025, and 612.001.118 respectively.

†University of Copenhagen, mia@di.ku.dk.

‡TU Eindhoven, mdberg@win.tue.nl, k.a.buchin@tue.nl, m.mehr@tue.nl, amehrabi@win.tue.nl.

[1] Throughout the paper, when we speak of cubes (or squares, or rectangles, or boxes) we always mean axis-aligned cubes (or squares, or rectangles, or boxes).

method applies includes the k-center problem in any L_p-metric, variants of the k-center problem where we want to minimize the sum (rather than maximum) of the cluster radii, and the 2-dimensional problem where we want to minimize the maximum or sum of the perimeters of the clusters. Our technique allows us, for instance, to answer rectilinear k-center queries in the plane in $O((1/\varepsilon)\log n + 1/\varepsilon^2)$ for $k = 2, 3$, in $O((1/\varepsilon)\log n + (1/\varepsilon^2)\,\mathrm{polylog}(1/\varepsilon))$ for $k = 4, 5$, and in $O((k/\varepsilon)\log n + (k/\varepsilon)^{O(\sqrt{k})})$ time for $k > 3$. We also show that for the rectilinear (or Euclidean) k-center problem, our method can be extended to deal with the capacitated version of the problem. In the capacitated version each cluster should not contain more than $\alpha \cdot (|P \cap \mathsf{Q}|/k)$ points, for a given $\alpha > 1$.

Some of the ideas underlying our approximate clustering queries can be used to efficiently compute a coreset for clustering in an offline setting [1]. As mentioned above, our method and all of its consequences will be deferred to the full version. Instead, in this note we present two exact linear-size data structures for rectilinear k-center queries in $\mathbb{R}^1$, and $\mathbb{R}^2$ when $k = 2$ or 3. Our results for this special case are mentioned in the abstract.

2 Exact k-Center Queries in $\mathbb{R}^1$

Given a set P of n points in $\mathbb{R}^1$ that we wish to preprocess into a data structure such that, given a query interval Q and a natural number $k \geqslant 2$, we can compute a set $\mathcal{C}$ of at most k intervals of the same length that together cover all points in $P_\mathsf{Q} := P \cap Q$ and whose length is minimum. The main result of this section is as follows.

Theorem 1 *Let a point set P of size n in $\mathbb{R}^1$ be given. There is a data structure that uses $O(n)$ storage such that, for a query range Q and a query value $k \geqslant 2$, we can answer a rectilinear k-center query in $O(\min\{k^2 \log^2 n, 3^k \log n\})$ time.*

The rest of the section is dedicated to the proof of the theorem. To present a proof, we present two data structures, which we call *the data structure for large k* and *the data structure for small k*. Both data structures are sorted arrays on the points $p_1, \ldots, p_n$ in P. Both query algorithms start by shrinking the query interval Q such that its left and right endpoint coincide with a point in P_Q.

A query algorithm for large k. This query algorithm uses a subroutine ***decider*** which, given an interval Q', a length L and an integer $\ell \leqslant k$, can decide in $O(\ell \log n)$ time if all points in $P \cap \mathsf{Q}'$ can be covered by ℓ intervals of length L. The global query algorithm then performs a binary search, using ***decider*** as subroutine, to find a pair of points $p_i, p_{i+1} \in P_\mathsf{Q}$ such that the first interval in an optimal solution covers p_i but not p_{i+1}. Then an optimal solution is found recursively for $k-1$ clusters within the query interval $\mathsf{Q} \cap [p_{i+1}, \infty)$. Next we describe the procedure ***decider***.

The procedure ***decider*** takes as input an integer ℓ, a number L, and an interval $\mathsf{Q}' = [a, a']$. It returns YES if Q' can be covered by at most ℓ subintervals of length L, and NO otherwise. The ***decider*** works as follows. Use binary search to find the first point $p_i \in P \cap \mathsf{Q}'$ not covered by the interval $[a : a + L]$, set $a := p_i$ and recurse. This continues until either all points in $P \cap \mathsf{Q}'$ are covered, or more than ℓ intervals are used. The ***decider*** runs in $O(\ell \cdot \log n)$ time and outputs YES in the first case and outputs NO in the latter case.

Now a given query interval $\mathsf{Q} := [x, x']$ and an integer k is handled as follows. Let $P_\mathsf{Q} := \{p_i, \ldots, p_j\}$, where the points are numbered from left to right. Thus $x = p_i$ and $x' = p_j$. We do a binary search on $\{p_i, \ldots, p_j\}$ to find the smallest index i^* with $i \leqslant i^* \leqslant j$ such that P_Q can be covered by k intervals of length $L := p_{i^*} - x$. Each decision in the binary search takes $O(k \log n)$ time by a call to ***decider***, so the whole binary search takes $O(k \log^2 n)$ time.

Let $\mathrm{OPT}_k(P)$ denote the minimum interval length needed to cover the points in a set P by k intervals. After finding i^* we know that

$$p_{i^*-1} - x < \mathrm{OPT}_k(P_\mathsf{Q}) \leqslant p_{i^*} - x.$$

If $\mathrm{OPT}_k(P_\mathsf{Q}) < p_{i^*} - x$, then the first interval in an optimal solution covers $\{p_i, \ldots, p_{i^*-1}\}$ and the remaining intervals cover $\{p_{i^*}, \ldots, p_j\}$. We now compute $\mathrm{OPT}_{k-1}(\{p_{i^*}, \ldots, p_j\})$ recursively, if $k - 1 > 1$ (if $k - 1 = 1$ then $\mathrm{OPT}_{k-1}(\{p_{i^*}, \ldots, p_j\})$ is clearly $\{p_{i^*}, \ldots, p_j\}$). If $\mathrm{OPT}_{k-1}(\{p_{i^*}, \ldots, p_j\}) < p_{i^*} - x$ then $\mathrm{OPT}_k(P_\mathsf{Q}) = \max\{p_{i^*-1} - x, \mathrm{OPT}_{k-1}(\{p_{i^*}, \ldots, p_j\})\}$, otherwise $\mathrm{OPT}_k(P_\mathsf{Q}) = p_{i^*} - x$.

It remains to analyze the running time of a query. The binary search takes $O(k \log^2 n)$ times, after which we do a recursive call in which the value of k has decreased by 1. (The problem is easily solved in $O(\log n)$ time when $k = 1$.) Hence the number of recursive call is k, leading to an $O(k^2 \log^2 n)$ query time, as claimed. Finding an optimal solution—and not just the value of an optimal solution—can be done within the same time bound. We get the following lemma.

Lemma 2 *Let P be a set of n points in $\mathbb{R}^1$. There is a data structure that uses $O(n)$ storage such that, for a query range Q and a query value $k \geqslant 2$, we can answer a rectilinear k-center query in $O(k^2 \log^2 n)$ time.*

A query algorithm for small k. Here we present an alternative solution, which is more efficient for small values of k. We begin with the following definition.

Definition 1 *Let P_Q be a set of points inside a query interval $Q = [x, x']$, such that $x, x' \in P_Q$. We call a point $r \in Q$ a* fair split point *if there is an optimal solution $\mathcal{C}_{opt}(Q) := \{I_1, I_2, \ldots, I_k\}$ for the k-center problem on P_Q such that (i) r does not lie in the interior of any subinterval $I_j \in \mathcal{C}_{opt}(Q)$, and (ii) the number of subintervals in $\mathcal{C}_{opt}(Q)$ lying to the left of r is $k(r-x)/(x'-x)$.*

Note that the split point r is not necessarily a point in P_Q, that is, it is not one of the given points. The following lemma is crucial in our analysis. We omit the proof of the lemma from this short version.

Lemma 3 *Let* Split$(Q) := \{s_1, s_2, \ldots, s_{k-1}\}$ *denote the set of points that partition Q into k equal-length subintervals. Then at least one of the points of* Split(Q) *is a fair split point.*

Lemma 3 suggests the following approach. A query with range $\mathsf{Q} = [x, x']$ and parameter k is answered as follows. Search the array for the successor $s(x)$ of x and the predecessor $p(x')$ of x' in P. Replace Q with $[s(x), p(x')]$, so that the left and right endpoints of the modified range Q are points from P. Partition Q into k equal-length subintervals. At each split point s_i of Q, recursively solve the problem on $\mathsf{Q}_{\text{left}} := [x, s_i]$ with parameter $k_{\text{left}} := i$ and on $\mathsf{Q}_{\text{right}} := [s_i, x']$ with parameter $k_{\text{right}} := k - i$. By Lemma 3, at (at least) one of the split points of Q the union of the returned intervals is an optimal solution. Moreover, we can easily maintain the best solution as we try all split points, so that after trying all split points we can return an optimal solution.

The recursion ends when $k = 1$. In this case we report $[s(x), p(x')]$ as the optimal solution. We obtain the following result.

Lemma 4 *Let P be a set of n points in $\mathbb{R}^1$. There is a data structure that uses $O(n)$ storage such that, for a query range Q and a query value $k \geqslant 2$, it can answer a rectilinear k-center query in $O(3^k \log n)$ time.*

Proof. It takes $O(\log n)$ time to find the successor and the predecessor of x and x' in P. Hence, we obtain the following recurrence for the time $T(k, n)$ needed to answer a k-center query on a point set of size n:

$$T(k,n) \leqslant \begin{cases} O(\log n) \\ O(\log n) + \sum_{i=1}^{k-1} T(i,n) + T(k-i,n) \end{cases}$$

where the first bound is for the case of $k = 1$ and the second bound is for $k > 1$. The recurrence solves to $T(n, k) = O(3^k \log n)$, and to see this, note that for the recurrence

$$T^*(k) = \sum_{i=1}^{k-1} T^*(i) + T^*(k-i)$$

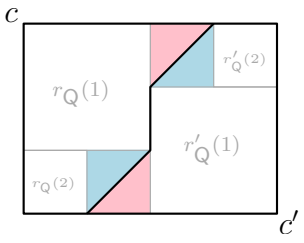


Figure 1: A query range Q and its regions. The boundary of Q and its ℓ_∞-bisector are in black. The regions $r_{\mathsf{Q}}(3)$ and $r'_{\mathsf{Q}}(3)$ are in light-blue and the regions $r_{\mathsf{Q}}(4)$ and $r'_{\mathsf{Q}}(4)$ are in pink.

we have

$$T^*(k) = 2\sum_{i=1}^{k-1} T^*(i) = 3T^*(k-1),$$

so with $T^*(1) = 1$ we obtain $T^*(k) = 3^{k-1}$, which implies $T(n, k) = O(3^k \log n)$. □

Proof of Theorem 1. Follows from Lemma 2 and Lemma 4.

3 Exact Rectilinear 2- and 3-Center Queries in $\mathbb{R}^2$

Given a set $P = \{p_1, p_2, \ldots, p_n\}$ of n points in $\mathbb{R}^2$ and an integer k. In this section we build a data structure $\mathcal{D}$ that stores the set P and, given an orthogonal query rectangle Q, can be used to quickly find an optimal solution for the k-center problem on P_{Q} for $k = 2$ or 3, where $P_{\mathsf{Q}} := P \cap \mathsf{Q}$.

2-center queries. We start by shrinking the query range Q such that each edge of Q touches at least one point of P. (The time for this step is subsumed by the time for the rest of the procedure.) It is well known that if we want to cover P_{Q} by two squares σ, σ' of minimum size, then σ and σ' both share a corner with Q and these corners are opposite corners of Q. We say that σ and σ' are *anchored* at the corner they share with Q. Thus we need to find optimal solutions for the two cases—σ and σ' are anchored at the topleft and bottomright corner of Q, or at the topright and bottomleft corner—and return the better one. Let c and c' be the topleft and the bottomright corners of Q, respectively. In the following we describe how to compute two squares σ and σ' of minimum size that are anchored at c and c', respectively, and whose union covers P_{Q}. The topright/bottomleft case can then be handled in the same way.

First we determine the ℓ_∞-bisector of c and c' inside Q; see Figure 1. The bisector partitions Q into two regions A and A', that respectively have c and c' on their boundary. Obviously in an optimal solution (of the type we are focusing on), the square σ must cover

$P_{\mathsf{Q}} \cap A$ and the square σ' must cover $P_{\mathsf{Q}} \cap A'$. To compute σ and σ', we thus need to find the points $q \in A$ and $q' \in A'$ with maximum ℓ_∞-distance to the corners c and c', respectively. To this end, we partition A and A' into subregions such that in each of the subregions the point with maximum ℓ_∞-distance to its corresponding corner can be found quickly via appropriate data structures discussed below. We assume w.l.o.g. that the x-span of Q is at least its y-span.

As Figure 1 suggests, we partition A and A' into subregions. We denote these subregions by $r_{\mathsf{Q}}(j)$ and $r'_{\mathsf{Q}}(j)$, for $1 \leqslant j \leqslant 4$. From now on we focus on reporting the point $q \in P$ in A with maximum ℓ_∞-distance to c; finding the farthest point to c' inside A' can be done similarly. Define four points $p(r_{\mathsf{Q}}(j)) \in P$ for $1 \leqslant j \leqslant 4$ as follows.

- The point $p(r_{\mathsf{Q}}(1))$ is the point of P_{Q} with maximum ℓ_∞-distance to c in $r_{\mathsf{Q}}(1)$. Note that this is either the point with maximum x-coordinate in $r_{\mathsf{Q}}(1)$ or the point with minimum y-coordinate.
- The point $p(r_{\mathsf{Q}}(2))$ is one of the bottommost points in $r_{\mathsf{Q}}(2)$.
- The point $p(r_{\mathsf{Q}}(3))$ is one of the bottommost points in $r_{\mathsf{Q}}(3)$.
- The point $p(r_{\mathsf{Q}}(4))$ is one of the rightmost point in $r_{\mathsf{Q}}(4)$.

Clearly $q = \arg\max_{1 \leqslant j \leqslant 4} \{d_\infty(p(r_{\mathsf{Q}}(j)), c)\}$, where d_∞ denotes the ℓ_∞-distance function.

Data structure. Our data structure now consists of the following components.

- We store P in a data structure $\mathcal{D}_1$ that allows us to report the extreme points in the x-direction and in the y-direction inside a rectangular query range. For this we use the structure by Chazelle [2], which uses $O(n \log^\varepsilon n)$ storage and has $O(\log n)$ query time.
- We store P in a data structure $\mathcal{D}_2$ with two components. The first component should answer the following queries: given a 45° query cone whose top bounding line is horizontal and that is directed to the left—we obtain such a cone when we extend the region $r_{\mathsf{Q}}(4)$ into an infinite cone—, report the rightmost point inside the cone. The second component should answer similar queries for cones that are the extension of $r_{\mathsf{Q}}(3)$.

  In the full version we describe a linear-size data structure that implements such a component and that has $O(\log n)$ query time.

Query procedure. Given Q, as we mentioned we shrink the query range and proceed as follows. Compute the ℓ_∞-bisector of Q. Query $\mathcal{D}_1$ with $r_{\mathsf{Q}}(1)$ and $r_{\mathsf{Q}}(2)$, respectively, to get the points $p(r_{\mathsf{Q}}(1))$ and $p(r_{\mathsf{Q}}(2))$. Then query $\mathcal{D}_2$ with u and u' to get the points $p(r_{\mathsf{Q}}(3))$ and $p(r_{\mathsf{Q}}(4))$, where u and u' are respectively the bottom and the top intersection points of ℓ_∞-bisector of Q and the boundary of Q. Among the at most four reported points, take the one with maximum ℓ_∞-distance to the corner c. This is the point $q \in P_{\mathsf{Q}} \cap A$ farthest from c.

Compute the point $q' \in P_{\mathsf{Q}} \cap A'$ farthest from c' in a similar fashion. Finally, report two minimum-size congruent squares σ, σ' anchored at c and c' and containing q and q', respectively.

Theorem 5 *Let P be a set of n points in $\mathbb{R}^2$. For any fixed $\varepsilon > 0$, there is a data structure using $O(n \log^\varepsilon n)$ storage that can answer rectilinear 2-center queries in $O(\log n)$ time.*

3-center queries. Given Q, it is easy to verify that at least one (out of the at most three) of the squares in the optimal solution must be anchored at one of the corners of Q. Following this observation, we try placing a square at a corner of Q and covering the remaining points in Q with at most two squares using the algorithm for 2-center queries. We do the same for the other corners of Q, and at the end we report the placement resulting in the best overall solution. We defer the details of the procedure to full version of the paper, and conclude the section with the following result.

Theorem 6 *Let P be a set of n points in $\mathbb{R}^2$. For any fixed $\varepsilon > 0$, there is a data structure using $O(n \log^\varepsilon n)$ storage that can answer rectilinear 3-center queries in $O(\log^2 n)$ time.*

References

[1] M. Abrahamsen, M. de Berg, K. Buchin, M. Mehr, and A. D. Mehrabi. A generic method for finding coresets for clustering problems. *To appear in EuroCG 2017.*

[2] B. Chazelle. A functional approach to data structures and its use in multidimensional searching. *SIAM J. Comput.* 17, pages 427–462, 1988.

[3] T. M. Chan. More planar two-center algorithms. *Comput. Geom. Theory Appl.* 13, pages 189–198, 1999.

[4] M. Hoffmann. A simple linear algorithm for computing rectilinear 3-centers. *Comput. Geom. Theory Appl.* 31, pages 150–165, 2005.

[5] M. Sharir and E. Welzl. Rectilinear and polygonal p-piercing and p-center problems. In *Proc. 12th ACM Symp. Comput. Geom. (SoCG)*, pages 122–132, 1996.

Delta-Fast Tries: Local Searches in Bounded Universes with Linear Space*

Marcel Ehrhardt† Wolfgang Mulzer†

Abstract

Let $U = \{0, 1, \ldots, 2^w - 1\}$ be a bounded universe. We present a dynamic data structure for predecessor searching in U that needs $O(\log\log\Delta)$ time for queries and $O(\log\log\Delta)$ expected time for updates, where Δ is the difference between the query element and its nearest neighbor in the structure. Our data structure requires linear space, improving a result by Bose *et al.*. It can be applied for answering approximate nearest neighbor queries in low dimensions.

1 Introduction

Predecessor searching is one of the oldest problems in theoretical computer science [3]. Let U be a totally ordered universe. The goal is to maintain a set $S \subseteq U$, $|S| = n$, while supporting *predecessor* and *successor* queries: given $q \in U$, find the largest element in S smaller than q (q's predecessor), or the smallest element in S larger than q (q's successor). In the *dynamic* version of the problem, we also want to be able to modify S by inserting and/or deleting elements.

In the *word-RAM model* of computation, all input elements are w-bit words, where $w \geq \log n$ is a parameter, and we are allowed to manipulate the input elements at the bit level. In this case, we may assume that the universe is $U = \{0, \ldots, 2^w - 1\}$. A classic solution for predecessor searching on the word-RAM is due to van Emde Boas, who described a dynamic data structure that requires space $O(n)$ and supports insertions and deletions in $O(\log\log U)$ time [6, 7].

In 2013, Bose *et al.* [2] described a word-RAM data structure for the predecessor problem that is *local* in the following sense. Let $q^+ := \min\{s \in S \mid s \geq q\}$ and $q^- := \max\{s \in S \mid s \leq q\}$ be the successor and predecessor of q. The structure by Bose *et al.* can answer predecessor and successor queries in $O(\log\log\Delta)$ time with $\Delta = \min\{|q - q^-|, |q - q^+|\}$. Their solution requires $\mathcal{O}(n \log\log\log|U|)$ words of space. Bose *et al.* apply their structure to obtain a fast data structure for approximate nearest neighbor queries in low dimensions.

We show how to reduce the space requirement to $O(n)$, while keeping the guarantees for the query times. This solves an open problem from [2], and also improves the space requirement for their nearest neighbor data structure. The full details can be found in the Master's thesis of the first author [5].

2 Preliminaries

Compressed Tries. Our data structure is based on *compressed tries* [3]. We interpret the elements from S as bitstrings of length w. The *trie* T' for S is a binary tree of height w. Each node $v \in T'$ corresponds to a bitstring $p_v \in \{0,1\}^*$. The root r has $p_r = \varepsilon$. For each inner node v, the left child u of v has $p_u = p_v 0$, and the right child w of v has $p_w = p_v 1$ (one of the two children may not exist). The bitstrings of the leaves correspond to the elements of S, and the bitstrings of the inner nodes are prefixes for the elements in S.

The *compressed trie* T for S is obtained from T' by contracting each maximal path of nodes with only one child into a single edge. Each inner node in T has exactly two children, and T has $O(n)$ nodes.

Let q be a bitstring of length at most w. The *longest common prefix* of q with S, $\mathrm{lcp}_S(q)$, is the longest prefix that q shares with an element in S. We say that q *lies on an edge* $e = (u, v)$ of T if p_u is a prefix of q and q is a proper prefix of p_v. If q lies on the edge (u, v), we call u the *least common ancestor* of q in T, denoted by $\mathrm{lca}_T(q)$. One can show that $\mathrm{lca}_T(q)$ is uniquely defined.

Associated Keys. Our algorithm uses the notion of *associated keys*. This notion was introduced for *z-fast tries* [1,10], and it is also useful in our data structure.

Associated keys provide a quick way to compute $\mathrm{lca}_T(q)$, for any element $q \in U$. A natural way to find $\mathrm{lca}_T(q)$ is to do binary search on the depth of $\mathrm{lca}_T(q)$: we initialize $(l, r) = [0, w]$ and let $m = (l + r)/2$. We denote by q_m the first m bits of q, and we check whether T has an edge $e = (u, v)$ such that q_m lies on e. If not, we set $r = m - 1$ and continue. Otherwise, we check if u is $\mathrm{lca}_T(q)$, by inspecting the other endpoint of e. If u is not $\mathrm{lca}_T(q)$, we set $l = m + 1$ and continue. In order to perform this search quickly, we need to find the edge e that contains a given prefix q_m. For this, we precompute for each edge e of T the first time that the binary search encounters a prefix that lies on e, and we let α_e be this prefix. We call α_e the *associated key* for $e = (u, v)$.

*Supported by DFG project MU/3501-1.

†Institut für Informatik, Freie Universität Berlin, Germany {marehr,mulzer}@inf.fu-berlin.de

The associated key can be computed in $O(1)$ as follows: consider the $\log w$-bit binary expansions $\ell_u = |p_u|_2$ and $\ell_v = |p_v|_2$ of the *lengths* of the prefixes p_u and p_v, and let ℓ' be the longest common prefix of ℓ_u and ℓ_v. Let ℓ be obtained by taking ℓ', followed by 1 and enough 0's to make a $\log w$-bit word. Let l be the number encoded by ℓ, and set α_e to the first l bits of p_v, see [5] for a detailed explanation.

Hash Maps. Our data structure also makes extensive use of hashing. In particular, we will maintain several succinct hashtables that store additional information for supporting fast queries. We will need the following theorem due to Demaine *et al.* [4].

Theorem 1 *For any $r \geq 1$, there exists a dynamic dictionary that stores entries with keys from U and with associated values of r bits each. The dictionary supports updates and queries in $O(1)$ time, using $O(n \log\log(|U|/n) + nr)$ bits of space. The bounds for the space and the queries are worst-case, the bounds for the updates hold with high probability.*

3 Static Δ-fast Tries

We begin by describing our data structure for the static case. The dynamic version will be discussed in the next section.

3.1 The Data Structure

Our data structure is organized as follows: We store S in a compressed trie T. The leaves of T are linked in sorted order. Furthermore, each node v of T stores pointers to the minimum and the maximum element in the subtree T_v of v. In addition to T, we maintain three hash maps H_Δ, H_z, and H_b.

We first describe the hash map H_Δ. Set $m = \log\log w$. For $i = 0, \ldots, m$, we let $h_i = 2^{2^i}$ and $d_i = w - h_i$. The hash map H_Δ stores the following information: for each $s \in S$ and each d_i, $i = 1, \ldots, m$, let $s_i = s_0 \ldots s_{d_i - 1}$ be the first d_i-bits of s and let $e = (u, v)$ be the edge of T such that s_i lies on e. Then, H_Δ stores the entry $H_\Delta[s_i] = u$.

Next, we describe the hash map H_z. It is defined similarly as the hash map used for z-fast tries [1, 10]. For each edge e of T, let α_e be the associated key of e (see Section 2). Then, H_z stores the entry $H_z[\alpha_e] = e$.

Finally, the hash map H_b is used to obtain linear space. It will be described below.

3.2 The Predecessor Query

Let $q \in U$ be the query, and let q^- and q^+ be the predecessor and the successor of q. We first show how to get a running time of $O(\log\log\Delta)$ for the queries, with $\Delta = |q - q^+|$. In Theorem 3, we will improve this to $\Delta = \min\{|q - q^-|, |q - q^+|\}$.

The predecessor search works in several *iterations*. In iteration i, we let q_i be the first d_i bits of q.

First, we check whether H_Δ contains an entry for q_i. If so, we know that T contains an edge e such that q_i lies on e, and that q_i is a prefix of $\mathrm{lcp}_S(q)$. We consider the two edges emanating from the lower endpoint of e, finding the e' that lies on the path to q (if the lower endpoint of e is $\mathrm{lca}_T(q)$, we are done). We take the associated key $\alpha_{e'}$ of e', and we use it to continue the binary search for $\mathrm{lca}_T(q)$, as described in Section 2. Since $|q_i| = d_i$, this binary search takes $O(\log(w - d_i)) = O(\log h_i)$ steps to complete. Once the lowest common ancestor $v = \mathrm{lca}_T(q)$ is available, we can find the predecessor of q in $O(1)$ additional time by inspecting the minimum and maximum elements in the subtrees for the two children of v and by using the pointers between the leaves in T. Details can be found in [5].

If H_Δ contains no entry for q_i and if q_i does not consist of all 1's, we check if H_Δ contains an entry for $q_i + 1$. Notice that $q_i + 1$ is the successor of q_i, e.g., if $q_i = 0000$, then $q_i + 1 = 0001$. If such an entry exists, first obtain $u = H_\Delta[q_i + 1]$, and the child v of u such that q_i lies on the edge $e = (u, v)$. Then, we find the minimum element in the subtree T_v. This is the successor q^+ of q. The predecessor q^- can be found by following the leaf pointers. This takes $O(1)$ time overall.

Finally, if both entries do not exist, we continue with iteration $i + 1$.

The total time for the predecessor query is $O(k + \log h_k)$, where k is the number of iterations and $\log h_k$ is the worst-case time for the predecessor search once one of the lookups in an iteration succeeds. By our predecessor algorithm, we know that S contains no element with prefix q_{k-1} or $q_{k-1} + 1$, but an element with prefix q_k or $q_k + 1$. Thus, we have $\Delta \geq 2^{h_{k-1}} = 2^{2^{2^{k-1}}}$, so $k \leq O(\log\log\log\Delta)$. Furthermore, since $h_k = (h_{k-1})^2$, it follows that $h_k = O(\log^2 \Delta)$.

3.3 Obtaining Linear Space

We now analyze the space requirement for our data structure. Clearly, the trie T and the hash map H_z require $O(n)$ words of space. Furthermore, as described so far, the space needed for H_Δ is $O(n \log\log w)$ words, since we store at most n entries for each height h_i, $i = 0, \ldots, m$.

Using a trick due to Pătraşcu [9], we can reduce the space requirement to linear. The idea is to store in H_Δ the *depth* d_u of each branch node u in T_Δ, instead of storing u itself. We then use an additional hash map H_b to obtain u.

This is done as follows: when trying to find the branch node u for a given prefix q_i, we first get the depth $d_u = |u|$ of u from H_Δ. After that, we look up the branch node $u = H_b[q_0 \ldots q_{d_u - 1}]$ from the hash

map H_b. Finally, we check whether u is actually the lowest branch node of q_i. If any of those steps fails, we return $\perp$.

Let us analyze the needed space: clearly, H_b needs $O(n)$ space, since it stores $O(n)$ entries. Furthermore, we have to store $O(n \log\log w)$ entries in H_Δ, each mapping a prefix q_i to the depth of its lowest branch node. This depth requires $\lceil \log w \rceil$ bits.

By Theorem 1, a retrieval only hash map for n' items and r bits of data is $O(n' \log\log \frac{|U|}{n'} + n'r)$ bits Therefore, the space for H_Δ is proportional to

$$n \log\log w \cdot \log\log \frac{|U|}{n \log\log w} + n \log\log w \cdot \lceil \log w \rceil$$
$$= O(n \log\log w \cdot \log w) = o(n \cdot w) \text{ bits},$$

where $n' = n \log\log w$, $r = \lceil \log w \rceil$ and $w = \log |U|$. Thus, we can store H_Δ in $O(n)$ words of w bits each. The following lemma summarizes the discussion

Lemma 2 *The Δ-fast trie needs $O(n)$ words space.*

3.4 Putting it Together

We can now obtain our result for the static predecessor problem.

Theorem 3 *The static Δ-fast trie solves the static predecessor problem with each operation in time $O(\log\log \min\{|q-q^-|, |q-q^+|\})$ and linear space. The preprocessing time is $O(n \log\log\log |U|)$ on sorted input.*

Proof. The normal search for $q \in S$ can be done in $O(1)$ time by a lookup in H_z. We have seen that the predecessor of q can be found in $O(\log\log |q - q^+|)$ time, and a symmetric result also holds for successor queries.

Similarly, we can achieve query time $O(\log\log |q - q^-|)$ by exchanging $H_\Delta[q_i - 1]$ with $H_\Delta[q_i + 1]$ in the query algorithm.

Therefore, by interleaving both searches, we obtain the desired running time of $O(\log\log \min\{|q-q^-|, |q-q^+|\})$. Of course, the pragmatic solution would be to add the case $H_\Delta[q_i - 1] \neq \perp$ to the query algorithm.

The preprocessing time is dominated by the time to fill the hash map H_Δ, since T and the hash maps H_z and H_b can be computed in linear time. Thus, the preprocessing time is $O(n \log\log\log |U|)$, because $\mathcal{O}(n \log\log w)$ nodes have to be inserted into H_Δ. The space requirement is linear (Lemma 2). □

4 Dynamic Δ-fast tries

We will now consider the dynamic case. For this, we need to explain how the update operations are implemented. Furthermore, we need a way to find and maintain the minimum and maximum element in each subtree of T. In the static case, this could be done by simply maintaining explicit pointers from each node $v \in T$ to the minimum and maximum element in T_v. In the dynamic case, we need a data structure which allows finding and updating these elements in in $O(\log\log \Delta)$ time.

4.1 Computing Lowest Common Ancestor

To perform the update operation, we need to be able to compute $\text{lca}_T(q)$ for any given element $q \in U$. For this, we will proceed as in the query algorithm from Section 3.2, but without the lookups for $H_\Delta[q_i - 1]$ and $H_\Delta[q_i + 1]$. By the analysis in Section 3.2, this will find $\text{lca}_T(q)$ in time $O(\log\log l)$, where l is height of $\text{lca}_T(q)$ in T.

Unfortunately, this height l might be as large as w. To get around this, we use a trick of Bose *et al.* [2]. Their idea is to perform a random shift of the universe. More precisely, we pick a random number $r \in U$, and we add r to all queries and update in the data structure (modulo $|U|$).

Lemma 4 (Lemma 4 in [2]) *After a random shift by r of U, the expected height of the lowest common ancestor of two fixed elements x and y is $O(\log |x-y|)$.*

Corollary 5 *Let $q \in U$, and let T be a randomly shifted Δ-fast trie. We can find $\text{lca}_T(q)$ in expected time $O(\log\log \Delta)$, where the expectation is over the random choice of the shift r.*

Proof. Set $x = q$, $y = q^+$ and let $\Delta = |q - q^+|$. We perform the doubly exponential search on the prefixes of q, as in Section 3.2 (without checking $q_i + 1$) to find the height h_k. After that, we resume the lowest common ancestor search on the remaining h_k bits. Since the number of remaining bits h_k is $O(\log \Delta)$ in expectation and by Jensen's inequality, the number of loop iterations k is $O(\log\log\log \Delta)$ in expectation. The expected running time is $k + \log h_k = O(\log\log \Delta)$. □

4.2 Managing the Minimum and Maximum Elements of the Subtrees

We also need to maintain the minimum and maximum elements in each subtree of T. In the static case, it suffices to have a pointer from a node to its minimal and maximal leaf, but in the dynamic case, we need an additional data structure.

For this, we use the fact that a given minimum (or maximum) leaf is common to at most w nodes of T. All these nodes form a subpath of a leaf-to-root path in T. Hence, if we maintain the nodes of this subpath in a concatenable queue data structure [8], we can obtain $O(\log w)$ update and query time to find the minimum (or maximum) element. However, we need

that the update and query time depend on the height h_i (i.e, the remaining bits) of the query. Thus, we partition the heights $\{0, 1, \ldots, w\}$ of a subpath into the sets $T_{-1} = \{0\}$, $T_i = [2^i, 2^{i+1})$, for $i = 0, \ldots, \log w - 1$, and $T_{\log w} = \{w\}$. Each of the sets is managed by a balanced binary tree, and all roots of those trees are linked together. The height of the i-th binary search tree is $\log |T_i| = O(i)$. Conversely, if a height h is given, the set $T_{\lfloor \log h \rfloor}$ is responsible for it.

Furthermore, T_{-1} is a leaf (the depth of that node is w) in the trie and therefore the minimum of the whole subpath. Thus, the minimum of a subpath can be found from a given node $v \in T_i$ in $O(i)$ time by following the pointers to the root of T_i and the pointers down to T_{-1}.

If a node v has $h_k = O(\log \Delta)$ remaining bits, the node is within the tree $T_{\lfloor \log h_k \rfloor}$. Thus, it takes $O(\log h_k) = O(\log \log \Delta)$ time to find the minimum. Furthermore, we can split and join the subpaths represented in this way in $O(\log h_k)$ time, where h_k is the height of the node where the operation occurs. Details can be found in [5].

4.2.1 Performing an Update

We know from the Lemma 4, that the lowest common ancestor has expected height $h_k = O(\log \Delta)$.

Lemma 6 *Inserting or deleting an element q into a Δ-fast trie takes $\mathcal{O}(\log \log \Delta)$ expected time, where the expectation is over the random choice of r.*

Proof. Inserting q into T splits an edge (u, v) of T into two edges (u, b) and (b, v). This creates two new nodes in T, a branch node and a leaf. The branch node is $\mathrm{lca}_T(q)$, and it has expected height $h_k = O(\log \Delta)$. So, finding it will take $O(\log \log \Delta)$ expected time, by Corollary 5.

The hash maps H_z and H_u can be updated in constant time. Now let us consider the update time of the hash map H_Δ. Remember that H_Δ stores the lowest branch nodes for all prefixes of the elements in S that have certain lengths. That means that all prefixes on the edge (b, v) which are stored in the hash map T_Δ need to be updated. Furthermore, prefixes at certain depths which are on the edge (b, q) need to be added. Note that for the edge (b, v), we will enumerate all prefixes at certain depths, but only select those that are on the edge. We will argue that a leaf-to-b path needs $O(\log \log \log \Delta)$ insertions or updates: We have to insert $Q_i := q_0 \ldots q_{d_i}$ for all $i = 1, \ldots$ until $d_i < |b|$. Since $d_i = w - h_i$, $|b| = w - O(\log \Delta)$ and $h_i = 2^{2^i}$, $c \log \Delta < 2^{2^i}$ when $i > \log \log(c \log \Delta)$. Thus, $i = \Theta(\log \log \log \Delta)$.

After that, the minimum and maximum elements for the subtrees of T have to be updated. This may update a subpath at a node of height $h_k = O(\log \Delta)$. As we have seen, this takes $O(\log h_k) = O(\log \log \Delta)$ time. Deletions are performed similarly. □

The following theorem summarizes our result.

Theorem 7 *Suppose we perform a random shift of U by r. Then, the Δ-fast trie solves the dynamic predecessor problem such that query operations take $O(\log \log \Delta)$ worst-case time and update operations take $O(\log \log \Delta)$ expected time. Here, $\Delta = \min\{|q - q^+|, |q - q^-|\}$, where $q \in U$ is the argument for the current operation and q^+ and q^- are the predecessor and successor of q in the current set. At any time, the data structure needs $O(n)$ words of space, where n is the size of the current set.*

References

[1] D. Belazzougui, P. Boldi, and S. Vigna. Dynamic z-fast tries. In *Proc. 17th Int. Symp. String Processing and Information Retrieval (SPIRE)*, pages 159–172, 2010.

[2] P. Bose, K. Douïeb, V. Dujmovic, J. Howat, and P. Morin. Fast local searches and updates in bounded universes. *Comput. Geom. Theory Appl.*, 46(2):181–189, 2013.

[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. MIT Press, third edition, 2009.

[4] E. D. Demaine, F. Meyer auf der Heide, R. Pagh, and M. Pătraşcu. De dictionariis dynamicis pauco spatio utentibus. In *Proc. 7th LATIN*, pages 349–361, 2006.

[5] M. Ehrhardt. An in-depth analysis of data structures derived from van-Emde-Boas-trees. Master's thesis, Freie Universität Berlin, 2015. `http://www.mi.fu-berlin.de/inf/groups/ag-ti/theses/download/Ehrhardt15.pdf`.

[6] P. van Emde Boas. Preserving order in a forest in less than logarithmic time and linear space. *Inform. Process. Lett.*, 6(3):80–82, 1977.

[7] P. van Emde Boas, R. Kaas, and E. Zijlstra. Design and implementation of an efficient priority queue. *Math. Systems Theory*, 10(2):99–127, 1976.

[8] F. P. Preparata and M. I. Shamos. *Computational geometry. An introduction*. Springer-Verlag, 1985.

[9] M. Pătraşcu. vEB space: Method 4. `http://infoweekly.blogspot.de/2010/09/veb-space-method-4.html`, 2010.

[10] M. Ružić. Making deterministic signatures quickly. *TALG*, 5(3):26:1–26:26, 2009.

A Simple Analysis of Rabin's Algorithm for Finding Closest Pairs*

Bahareh Banyassady† Wolfgang Mulzer†

Abstract

The closest-pair problem is one of the most basic topics in computational geometry: given a set $P \subset \mathbb{R}^2$ of n points in the plane, find two distinct points $p, q \in P$ that minimize the Euclidean distance $d(p, q)$, among all pairs of points in P. In the algebraic decision tree model, this problem can be solved optimally in $\Theta(n \log n)$ time.

However, already in 1976, Rabin observed in a seminal work that, using the floor function and randomization, this can be improved to $O(n)$ *expected* time. We provide a new and simplified analysis of Rabin's algorithm that is intended to make this result more accessible to the modern reader.

1 Introduction

The closest-pair problem in computational geometry is defined as follows: given a set $P \subset \mathbb{R}^d$ of n points in d dimensions, find two distinct points $p, q \in P$ that minimize the Euclidean distance $d(p, q)$, among all pairs of points in P. As was already observed in the 1970s, this problem can be solved in the plane in $O(n \log n)$ time by computing the Delaunay triangulation of P [11, 13, 14]. For any fixed $d \geq 2$, the classic divide-and-conquer algorithm by Bentley and Shamos also achieves $O(n \log n)$ time [2, 5]. This running time is asymptotically optimal in the algebraic decision tree model of computation [1, 11].

However, this is far from the whole story. Once we leave the confines of the algebraic decision tree model, faster algorithms are possible. For example, in the *transdichotomous* model, where the input may be manipulated at the bit-level, we can compute planar Delaunay triangulations, and hence also the planar closest pair, in $O(n \log \log n)$ expected time [3].

More famously, if we admit the *floor-function* $x \mapsto \lfloor x \rfloor$ into our model, there are randomized algorithms that can compute the closest pair in *linear* expected time. This was shown first by Rabin, in a famous paper that is often considered the starting point for the study of randomized algorithms [12]. Indeed, it has been claimed that Rabin's algorithm is one of the first randomized algorithms in theoretical computer science [15].

*Supported by DFG project MU/3501-2.

†Institut für Informatik, Freie Universität Berlin, Germany
{bahareh, mulzer}@inf.fu-berlin.de

Since then, a lot more work has been done on the closest pair problem: Dietzfelbinger *et al.* [6] describe how to implement rigorously the hashing-based data structure that was left open in Rabin's original algorithm. They also provide a detailed analysis of Rabin's algorithm that shows that bounded independence suffices to obtain the desired expected running time. Khuller and Matias [9] describe an alternative, sieve based approach to closest pairs, and Golin *et al.* [7] give a very simple randomized algorithm that uses the randomized incremental construction paradigm and that can be found in several textbooks [8, 10]. Finally, Chan [4] presents a randomized framework for geometric optimization problems that also leads to a new randomized linear-time algorithm for the closest pair problem. The survey by Smid [15] contains a much more comprehensive treatment of these results.

Despite the amount of activity on the closest pair problem, the presentation of Rabin's original algorithm has remained untouched for more than 40 years [12]. We give a new description and analysis of this algorithm in today's terms. We hope that this simplified presentation will make Rabin's algorithm more accessible for modern students of computational geometry, and it may lead to new insights on the closest pair problem. Here, we focus on the planar case, although all the arguments hold for d dimensions, when $d \in \mathbb{N}$ is a constant.

2 Preliminaries

Let $P \subset \mathbb{R}^2$ be a set of n points in the plane such that all $\binom{n}{2}$ interpoint distances in P are pairwise distinct. Furthermore, we assume that P lies completely in the upper right quadrant, i.e., that all points in P have positive coordinates.

For $i \in \mathbb{Z}$, we define the *ith grid* $\mathcal{G}_i$ as the subdivision of the plane into square grid *cells* of diameter 2^i. The cells have pairwise disjoint interiors, side length $2^i/\sqrt{2}$, and they cover the whole plane. The grid $\mathcal{G}_i$ is aligned such that the origin appears as a corner of four adjacent grid cells. The *neighborhood* of a grid cell $\sigma \in \mathcal{G}_i$ consists of the 7×7 square of grid cells centered at σ. Two cells $\sigma, \tau \in \mathcal{G}_i$ are *neighboring* if τ lies in the neighborhood of σ (and hence σ lies in the neighborhood of τ). We define the identifier of a cell $\sigma \in \mathcal{G}_i$ as a pair from $\mathbb{Z} \times \mathbb{Z}$, indicating the column and row for σ. The cell whose lower left corner is the

origin is identified as $(0,0)$. As it is standard in randomized algorithms for the closest pair problem, we assume that there is an operation **findGridCell**(i,p), for $i \in \mathbb{Z}$ and $p \in \mathbb{R}^2$, that returns the identifier of the cell σ in $\mathcal{G}_i$, that contains the point p [7, 9, 12, 15]. Using the floor function, **findGridCell** can be implemented in $O(1)$ time.

A *cell dictionary* is a data structure for storing cells that are in a grid [7]. It supports the following operations:

- **create**(i): create an empty cell dictionary D for $\mathcal{G}_i$.
- **insert**(D,p): insert the point p into D.
- **lookup**(D,σ): Suppose that D is a cell dictionary for $\mathcal{G}_i$ and σ is a cell in $\mathcal{G}_i$. Then, the lookup operation reports the set of all points stored in D that lie in σ. It returns $\emptyset$, if σ contains no points.

As explained by Golin *et al.* [7], the cell dictionary can be implemented using the hashing-based techniques of Dietzfelbinger *et al.* [6], so that the expected time for **create** and **insert** is $O(1)$ and the expected time for **lookup** is $O(1+k)$, where k is the output size. Alternatively, an implementation based on binary search trees achieves a worst-case running time of $O(1)$ for **create**, $O(\log n)$ for **insert**, and $O(\log n + k)$ for **lookup**, where again k denotes the output size. In our analysis, we will separately count the operations on the cell dictionary and the remaining computational steps.

3 The Algorithm

We now describe our version of Rabin's algorithm. Let $P \subset \mathbb{R}^2$ be the n input points in the plane, and set $k = \lfloor \log n \rfloor - 1$. We compute a *random gradation* $P = P_0 \supset P_1 \supset P_2 \supset \cdots \supset P_k$ of P, where for $i = 1, \ldots, k$, the set P_i is a random subset of P_{i-1} with exactly $|P_i| = \lfloor n/2^i \rfloor$ elements. In particular, we have $|P_k| = O(1)$.

The algorithm proceeds in *rounds*. The rounds are numbered from $k+1$ to 1, beginning with round $k+1$. For $i = k+1, \ldots, 1$, the goal of round i is to compute a cell dictionary D_{i-1} that stores all points from P_{i-1} such that:

(A) each cell in D_{i-1} contains at most one point from P_{i-1}; and

(B) let $p, q \in P_{i-1}$ be the two points that constitute the closest pair in P_{i-1}. Then, the cells in D_{i-1} that contain p and q are neighboring.

Since $|P_k| = O(1)$, this can be easily achieved in round $k+1$: by checking all pairs in P_k, we compute the closest pair distance δ_k for P_k. Then, we set $j = \lceil \log \delta_k \rceil - 1$, and we create a cell dictionary D_k for the grid $\mathcal{G}_j$. We insert all points of P_k into D_k. Since the diameter of the cells of $\mathcal{G}_j$ is $2^j \in [\delta_k/2, \delta_k)$, each cell in D_k contains at most one point of P_k. Furthermore, since the cells in $\mathcal{G}_j$ have side length at least $\delta_k/2\sqrt{2}$, the cells for the closest pair in P_k are neighboring.

In round i, $i = k, \ldots, 1$, the algorithm has the cell dictionary D_i from the previous round available, and it constructs the dictionary D_{i-1} for round i as follows: first, we insert all points from P_{i-1} into D_i. Then, for each non-empty cell σ in D_i, we find the set Q_σ of points inside σ. We use a brute-force algorithm to compute the closest pair distance δ_σ for Q_σ, and we set $\delta_{i-1} = \min_{\sigma \in D_i} \delta_\sigma$. Next, we set $j = \lceil \log \delta_{i-1} \rceil - 1$, and we create a cell dictionary D_{i-1} for $\mathcal{G}_j$. Then, we insert all points from P_{i-1} into D_{i-1}. By construction, the diameter of the cells of D_{i-1} is $2^j \in [\delta_{i-1}/2, \delta_{i-1})$, and so, each cell of D_{i-1} contains at most one point of P_{i-1}. Furthermore, since the cells in D_{i-1} have side length at least $\delta_{i-1}/2\sqrt{2}$, the cells for the closest pair in P_{i-1} must be neighboring (we note that the closest pair distance could be much less than δ_{i-1}, since we only check the distances inside each cell of D_i to compute δ_{i-1}). The following lemma summarizes the running time of round i.

Lemma 1 *Let $i \in \{1, \ldots, k\}$. In round i, the algorithm performs $O(|P_{i-1}|)$ cell dictionary operations, and the additional work is proportional to*

$$\sum_{\sigma \in D_i} |Q_\sigma|^2,$$

where the sum is over all non-empty cells σ stored in D_i, and Q_σ is defined as $P_{i-1} \cap \sigma$.

Once we have the cell dictionary D_0 for P at hand, we can compute the closest pair of P with $O(n)$ cell dictionary operations and $O(n)$ additional work: we simply check the neighborhood of each non-empty cell in D_0, and we find the closest pair among all points that reside in these cells. Since each cell of D_0 contains at most one point, and since the closest pair must be in neighboring cells, this gives the closest pair of P in the desired time.

4 Analysis

We now analyse our version of Rabin's algorithm. For $i \in \{1, \ldots, k\}$, let

$$Z_i = \sum_{\sigma \in D_i} |Q_\sigma|^2$$

be the random variable that represents the amount of work in round i, excluding the time for the cell dictionary operations. We will show that $\mathbf{E}[Z_i] = O(|P_{i-1}|)$, for $i = 1, \ldots, k$.

For this, we fix an $i \in \{1, \ldots, k\}$ and a subset $Q \subseteq P$ with $|Q| = \lfloor n/2^{i-1} \rfloor$. First, we rewrite Z_i in a slightly different way.

Lemma 2 *Let $i \in \{1, \ldots, k\}$ and $Q \subseteq P$ with $|Q| = \lfloor n/2^{i-1} \rfloor$ be given. For $p \in Q$, let X_p denote the number of points from Q that are in the same cell of D_i as p (including p). Then,*

$$\mathbf{E}[Z_i \mid P_{i-1} = Q] = \sum_{p \in Q} \mathbf{E}[X_p \mid P_{i-1} = Q].$$

Proof. This can be seen through a simple application of the double-counting principle. We have

$$\begin{aligned}
&\mathbf{E}[Z_i \mid P_{i-1} = Q] \\
&= \mathbf{E}\Big[\sum_{\sigma \in D_i} |Q_\sigma|^2 \;\Big|\; P_{i-1} = Q\Big] \\
&= \mathbf{E}\Big[\sum_{\sigma \in D_i} \sum_{p \in \sigma \cap Q} |Q_\sigma| \;\Big|\; P_{i-1} = Q\Big] \\
&= \mathbf{E}\Big[\sum_{p \in Q} \sum_{\substack{\sigma \in D_i \\ p \in \sigma}} |Q_\sigma| \;\Big|\; P_{i-1} = Q\Big] \\
&= \mathbf{E}\Big[\sum_{p \in Q} X_p \;\Big|\; P_{i-1} = Q\Big],
\end{aligned}$$

as claimed. □

Next, we bound the expectation of X_p.

Lemma 3 *Let $i \in \{1, \ldots, k\}$ and $Q \subseteq P$ with $|Q| = \lfloor n/2^{i-1} \rfloor$ be given. Let $p \in Q$, and let X_p denote the number of points from Q that are in the same cell of D_i as p (including p). Then,*

$$\mathbf{E}[X_p \mid P_{i-1} = Q] = O(1).$$

Proof. Set $q = |Q|$, and let $r \in \{1, \ldots, q\}$. We know that $q \geq 2$ (since $|P_{i-1}| \geq 2$). First, we show that

$$\Pr[X_p \geq r \mid P_{i-1} = Q] \leq 2r\Big(\frac{2}{3}\Big)^{r-1}. \qquad (1)$$

For this, consider the event that $X_p \geq r$. This means that, the cell $\sigma \in D_i$ that contains p must contain at least r points from Q. How can this happen? For $j \in \mathbb{Z}$, let τ_j be the cell of $\mathcal{G}_j$ that contains p, and let $Q_j = Q \cap \tau_j$. Obviously, for j small enough, we have $Q_j = \{p\}$, for j large enough, we have $Q_j = Q$ (since we assumed that all coordinates in P are positive), and as j increases, Q_j grows monotonically. Let j^* be the smallest index such that τ_{j^*} has at least r point of Q , $|Q_{j^*}| \geq r$. And let $R \subseteq Q_{j^*}$ be an arbitrary subset with $|R| = r$. Now, if the cell $\sigma \in D_i$ with $p \in \sigma$ contains at least r points from Q, due to the definition of j^*, the grid cell τ_{j^*} is a subcell of the grid cell σ, and so, σ contains all of R. Furthermore, since σ appears in D_i, by the invariant it must be the case that σ contains at most one point from P_i; see Figure 1. Thus, $|P_i \cap R| \leq 1$, which implies:

$$\Pr[X_p \geq r \mid P_{i-1} = Q] \leq \Pr[|P_i \cap R| \leq 1 \mid P_{i-1} = Q].$$

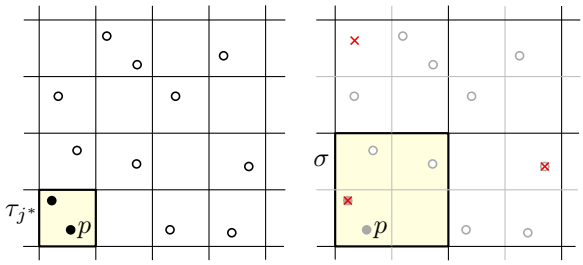


Figure 1: Left: The grid $\mathcal{G}_{j^*}$ and the set Q of points are shown. For $q = 2$ and the point p, the cell τ_{j^*} is specified. The points of the set $R \subseteq Q_{j^*}$, are filled with black color. Right: The cell σ of D_i is specified. The red crosses show the points of P_i. The cell σ contains all the points of R, thus $|P_i \cap R| = 1$.

Given $P_{i-1} = Q$, we have that P_i is a random subset of $\lfloor q/2 \rfloor$ points from Q. Since R is also a subset of Q, the desired probability is easily bounded. If $r > \lceil q/2 \rceil$, then the intersection of P_i and R is not empty, in another words $\Pr[|P_i \cap R| = 0 \mid P_{i-1} = Q] = 0$. Otherwise, if $r \leq \lceil q/2 \rceil$, we have

$$\begin{aligned}
&\Pr[|P_i \cap R| = 0 \mid P_{i-1} = Q] \\
&= \binom{q-r}{\lfloor q/2 \rfloor} \Big/ \binom{q}{\lfloor q/2 \rfloor} \\
&= \frac{(q-r)!}{\lfloor q/2 \rfloor! \, (\lceil q/2 \rceil - r)!} \frac{\lfloor q/2 \rfloor! \, \lceil q/2 \rceil!}{q!} \\
&= \prod_{k=0}^{r-1} \frac{\lceil q/2 \rceil - k}{q-k} \leq \left(\frac{\lceil q/2 \rceil}{q}\right)^r \leq \Big(\frac{2}{3}\Big)^r,
\end{aligned}$$

since $q \geq 2$. Moreover, if we have $r > \lceil q/2 \rceil + 1$, then $\Pr[|P_i \cap R| = 1 \mid P_{i-1} = Q] = 0$. Otherwise, if $r \leq \lceil q/2 \rceil + 1$, we have

$$\begin{aligned}
&\Pr[|P_i \cap R| = 1 \mid P_{i-1} = Q] \\
&= r \binom{q-r}{\lfloor q/2 \rfloor - 1} \Big/ \binom{q}{\lfloor q/2 \rfloor} \\
&= r \frac{(q-r)!}{(\lfloor q/2 \rfloor - 1)! \, (\lceil q/2 \rceil - r + 1)!} \frac{\lfloor q/2 \rfloor! \, \lceil q/2 \rceil!}{q!} \\
&= r \frac{\lfloor q/2 \rfloor}{q - r + 1} \prod_{k=0}^{r-2} \frac{\lceil q/2 \rceil - k}{q-k} \\
&\leq r \left(\frac{\lceil q/2 \rceil}{q}\right)^{r-1} \leq r\Big(\frac{2}{3}\Big)^{r-1},
\end{aligned}$$

since $q \geq 2$ and $\lfloor q/2 \rfloor / (q - r + 1) \leq 1$. Now, (1) follows, since

$$\begin{aligned}
\Pr[X_p \geq r \mid P_{i-1} = Q] &\leq \Pr[|P_i \cap R| \leq 1 \mid P_{i-1} = Q] \\
&= \Pr[|P_i \cap R| = 0 \mid P_{i-1} = Q] \\
&\quad + \Pr[|P_i \cap R| = 1 \mid P_{i-1} = Q] \\
&\leq \Big(\frac{2}{3}\Big)^r + r\Big(\frac{2}{3}\Big)^{r-1} \\
&\leq 2r\Big(\frac{2}{3}\Big)^{r-1}.
\end{aligned}$$

Now, we have

$$\begin{aligned}\mathbf{E}[X_p \mid P_{i-1} = Q] &\leq \sum_{r=1}^{q} \Pr[X_p \geq r \mid P_{i-1} = Q] \\ &\leq \sum_{r=1}^{\infty} 2r\Big(\frac{2}{3}\Big)^{r-1} = O(1),\end{aligned}$$

as claimed. □

Lemma 4 *For $i \in \{1, \ldots, k\}$, $\mathbf{E}[Z_i] = O(|P_{i-1}|)$.*

Proof. Fix $Q \subseteq P$ with $|Q| = \lfloor n/2^{i-1} \rfloor$. By Lemma 2 and Lemma 3, we have

$$\begin{aligned}\mathbf{E}[Z_i \mid P_{i-1} = Q] &= \sum_{p \in Q} \mathbf{E}[X_p \mid P_{i-1} = Q] \\ &= \sum_{p \in Q} O(1) = O(|Q|) = O(|P_{i-1}|).\end{aligned}$$

Thus, using the law of total probability

$$\begin{aligned}E[Z_i] &= \sum_{\substack{Q \subseteq P \\ |Q| = \lfloor n/2^{i-1} \rfloor}} \Pr[P_{i-1} = Q]\mathbf{E}[Z_i \mid P_{i-1} = Q] \\ &= O(|P_{i-1}|) \sum_{\substack{Q \subseteq P \\ |Q| = \lfloor n/2^{i-1} \rfloor}} \Pr[P_{i-1} = Q] \\ &= O(|P_{i-1}|),\end{aligned}$$

as claimed. □

The following theorem summarizes the analysis:

Theorem 5 *The algorithm from Section 3 computes the closest pair for P in expected time $O(n)$.*

Proof. We already argued correctness in Section 3. As mentioned above, using randomization and the floor function, we can implement all operations of the cell dictionary in $O(1)$ expected time [6, 7]. Thus, by Lemma 1, the total expected time for the cell dictionary operations is:

$$\sum_{i=1}^{k} O\left(|P_{i-1}|\right) = O\left(\sum_{i=0}^{k-1} \frac{n}{2^i}\right) = O(n).$$

Similarly, by Lemma 1 and Lemma 4, the total expected time for the remaining steps is:

$$\sum_{i=1}^{k} \mathbf{E}[Z_i] = O\left(\sum_{i=1}^{k} |P_{i-1}|\right) = O(n).$$

We also argued that, having the cell dictionary D_0 for P, we can compute the closest pair of P in $O(n)$ time. This concludes the analysis. □

References

[1] M. Ben-Or. Lower bounds for algebraic computation trees (preliminary report). In *Proc. 15th Annu. ACM Sympos. Theory Comput. (STOC)*, pages 80–86, 1983.

[2] J. L. Bentley and M. I. Shamos. Divide-and-conquer in multidimensional space. In *Proc. 8th Annu. ACM Sympos. Theory Comput. (STOC)*, pages 220–230, 1976.

[3] K. Buchin and W. Mulzer. Delaunay triangulations in $o(\text{sort}(n))$ time and more. *J. ACM*, 58(2):6, 2011.

[4] T. M. Chan. Geometric applications of a randomized optimization technique. *Discrete Comput. Geom.*, 22(4):547–567, 1999.

[5] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. MIT Press, third edition, 2009.

[6] M. Dietzfelbinger, T. Hagerup, J. Katajainen, and M. Penttonen. A reliable randomized algorithm for the closest-pair problem. *J. Algorithms*, 25(1):19–51, 1997.

[7] M. Golin, R. Raman, C. Schwarz, and M. Smid. Simple randomized algorithms for closest pair problems. *Nordic J. Comput.*, 2(1):3–27, 1995.

[8] S. Har-Peled. *Geometric approximation algorithms*. Mathematical Surveys and Monographs. American Mathematical Society, vol 173, 2011.

[9] S. Khuller and Y. Matias. A simple randomized sieve algorithm for the closest-pair problem. *Inform. and Comput.*, 118(1):34–37, 1995.

[10] J. M. Kleinberg and É. Tardos. *Algorithm design*. Addison-Wesley, 2006.

[11] F. P. Preparata and M. I. Shamos. *Computational geometry*. Springer-Verlag, 1985.

[12] M. O. Rabin. Probabilistic algorithms. In *Algorithms and Complexity: New Directions and Recent Results*, pages 21–40. Academic Press, 1976.

[13] M. I. Shamos. Geometric complexity. In *Proc. 7th Annu. ACM Sympos. Theory Comput. (STOC)*, pages 224–233, 1975.

[14] M. I. Shamos and D. Hoey. Closest-point problems. In *Proc. 16th Annu. IEEE Sympos. Found. Comput. Sci. (FOCS)*, pages 151–162, 1975.

[15] M. Smid. Closest-point problems in computational geometry. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 877–935. Elsevier, 2000.

A generalization of crossing families

Patrick Schnider*

Abstract

For a set $\mathcal{P}$ of points in the plane, a *crossing family* for $\mathcal{P}$ is a set $\mathcal{C}$ of line segments, each joining two of the points from $\mathcal{P}$, such that any two line segments from $\mathcal{C}$ cross. We investigate the following generalization of crossing families: a *spoke set* for $\mathcal{P}$ is a set of lines such that each unbounded region of the induced line arrangement contains at least one point of $\mathcal{P}$.

We show that every point set of size n has a spoke set of size $\sqrt{\frac{n}{8}}$. We also characterize the matchings obtained by selecting exactly one point in each unbounded region and connecting every such point to the point in the antipodal unbounded region.

1 Introduction

Let $\mathcal{P}$ be a finite point set in *general position* (i.e., no three points on a line). Throughout this paper, we assume all point sets to be in general position. A *crossing family* for $\mathcal{P}$ is a set $\mathcal{C}$ of line segments, each joining two of the points from $\mathcal{P}$, such that any two line segments from $\mathcal{C}$ cross (i.e., intersect in their interior). Crossing families were introduced by Aronov et al. [1], who have shown that any set of n points in general position has a crossing family of size $\sqrt{\frac{n}{12}}$. Since then, there have been several results about crossing families [3, 4], but even though it is conjectured that any point set in general position has a crossing family of linear size [1], the bound of Aronov et al. is still the best known result.

A point set $\mathcal{A}$ *avoids* a point set $\mathcal{B}$ if no line through two points in $\mathcal{A}$ intersects the convex hull of $\mathcal{B}$. Note that this means that every point in $\mathcal{B}$ sees the points in $\mathcal{A}$ in the same rotational order. If $\mathcal{B}$ also avoids $\mathcal{A}$, the two sets are called *mutually avoiding*. The bound in [1] on the size of the largest crossing family is proven in two steps: first it is shown that two mutually avoiding sets $\mathcal{A}$ and $\mathcal{B}$, each of size k, induce a crossing family of size k. Then it is shown that every set of n points in general position contains two mutually avoiding subsets of size $\sqrt{\frac{n}{12}}$. In this paper we will follow the same approach, but for a generalization of crossing families.

Bose et al. [2] have introduced the following generalization of crossing families: A *spoke set of size k* for $\mathcal{P}$ is a set $\mathcal{S}$ of k pairwise non-parallel lines such that in each unbounded region of the arrangement defined by the lines in $\mathcal{S}$ there lies at least one point of $\mathcal{P}$. Note that it is easy to obtain a spoke set from a crossing family by slightly rotating the supporting lines of the line segments in the crossing family. Then each endpoint of a line segment in the crossing family lies in a different unbounded region. We will show that every set of n points in general position contains a spoke set of size $\sqrt{\frac{n}{8}}$. To this end, we first translate the notion of spoke sets to the dual setting in Section 2. In Section 3 we then use the dual version to construct large spoke sets for the union of two point sets $\mathcal{A}$ and $\mathcal{B}$, where $\mathcal{A}$ avoids $\mathcal{B}$ and $\mathcal{A}$ and $\mathcal{B}$ can be separated by a line. Finally, we show that every point set contains such point sets $\mathcal{A}$ and $\mathcal{B}$ and give bounds on their sizes.

The motivation for the introduction of spoke sets in [2] is the fact that with a spoke set of size k for $\mathcal{P}$, one can construct a covering of the edge set of the complete geometric graph drawn on $\mathcal{P}$ with $n-k$ crossing-free spanning trees. The result in this paper thus also improves the previous upper bound of $n - \sqrt{\frac{n}{12}}$ for this problem. However, the original question from [2], whether there is always a spoke set of linear size, remains open.

Another interesting question is whether it is always possible to find a crossing family of size linear in the size of the largest spoke set. Theorem 6 is a first step in this direction as it characterizes the matchings obtained from spoke sets and shows that even though they might not all be crossing families, they still satisfy a number of conditions.

For space reasons, we will not be able to give all proofs. Instead, we refer the interested reader to full version [6].

Preliminaries

Let $\mathcal{S}$ be a spoke set of size k for $\mathcal{P}$. Consider the ordering of $\mathcal{S} = \{\ell_1, \dots, \ell_k\}$ by increasing slope. Let U_i^+ be the unbounded region that lies below $\ell_1, \dots, \ell_i$ and above $\ell_{i+1}, \dots \ell_k$. Similarly let U_i^- be the unbounded region that lies above $\ell_1, \dots, \ell_i$ and below $\ell_{i+1}, \dots \ell_k$. We call the regions U_i^+ and U_i^- *antipodal*.

Let $\mathcal{Q}$ be a subset of $\mathcal{P}$ that has exactly one point in each unbounded region. Note that then each line of $\mathcal{S}$ is a halving line for $\mathcal{Q}$. The *spoke matching* of $\mathcal{Q}$ is the matching obtained by drawing a straight line segment from each point p in $\mathcal{Q}$ to the unique point

*Department of Computer Science, ETH Zürich, patrick.schnider@inf.ethz.ch

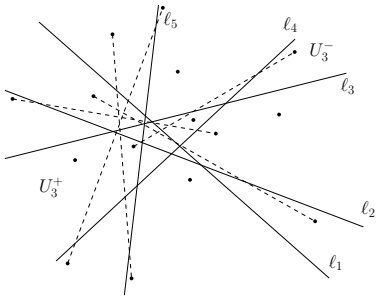


Figure 1: A spoke set and a spoke matching (dashed).

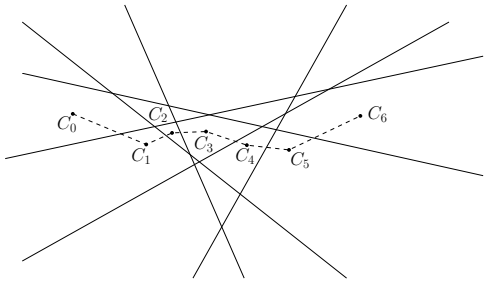


Figure 2: A line-monotone AB-semialternating cell-path of length 6.

q in $\mathcal{Q}$ that lies in the antipodal unbounded region of the spoke set. See Figure 1 for an example. Note that in a spoke matching, each edge intersects every line of the spoke set. In Section 4 we characterize the geometric matchings that are spoke matchings.

2 Spoke sets under duality

In this section we will translate the properties of spoke sets into the dual setting, that is under the point-line duality. For this we start with some definitions.

Given an arrangement $\mathcal{A}$ of lines, without loss of generality none of them horizontal or vertical, a *cell-path* R is a sequence of cells such that consecutive cells share an edge. If the edge shared by two consecutive cells is a subset of some line a_i of $\mathcal{A}$, we say that R *crosses* a_i. The *length* of a cell-path is one less than the number of cells involved. We call a cell-path *line-monotone* if it crosses each line of $\mathcal{A}$ at most once.

If $\mathcal{A}'$ is an arrangement induced by a subset of the lines of $\mathcal{A}$, then R *restricted to* $\mathcal{A}'$ is the cell path obtained by replacing each cell C of $\mathcal{A}$ in R by the cell C' in $\mathcal{A}'$ with $C \subseteq C'$ and deleting consecutive multiples.

Finally, for a cell-path $R = (C_0, C_1, \ldots, C_k)$, let a_i be the line in $\mathcal{A}$ that contains the edge shared by C_i and C_{i+1}. We call the pair (a_{2j}, a_{2j+1}) *AB-alternating*, if C_{2j+1} either lies above both a_{2j} and a_{2j+1} or below both. We call a cell path $P = (C_0, C_1, \ldots, C_{2k})$ *AB-semialternating* if for every $j < k$ the pair (a_{2j}, a_{2j+1}) is AB-alternating. See Figure 2 for an example.

We now have all the vocabulary that is necessary to describe the dual of spoke sets: given an arrangement $\mathcal{A}$ of lines, a *spoke path* $(R, \mathcal{A}')$ is a cell-path R together with an arrangement $\mathcal{A}'$ induced by a subset of the lines of $\mathcal{A}$, such that R restricted to $\mathcal{A}'$ is line-monotone and AB-semialternating. The length of a spoke path $(R, \mathcal{A}')$ is the length of R restricted to $\mathcal{A}'$. Note that all the definitions generalize to x-monotone pseudoline arrangements.

Lemma 1 *Let $\mathcal{P}$ be a point set and $\mathcal{P}^*$ its dual line arrangement. Then $\mathcal{P}$ contains a spoke set of size k if and only if $\mathcal{P}^*$ contains a spoke path of length $2k$.*

For a proof we refer to the full version. It is worth mentioning that for a spoke path $(R, \mathcal{A}')$, the primal of $\mathcal{A}'$ corresponds to a subset of $\mathcal{P}$ that has exactly one point in each unbounded region. The fact that all the points in the primal of $\mathcal{A}'$ are in unbounded regions follows from the line-monotonicity of R restricted to $\mathcal{A}'$. The AB-semialternation implies that two lines a_{2j} and a_{2j+1} correspond to endpoints of the spoke matching in the primal.

3 Finding large spoke sets

In this section, we will construct large spoke sets by constructing long spoke paths in the dual arrangement.

Lemma 2 *Let $\mathcal{A}$ and $\mathcal{B}$ be two disjoint point sets of size k such that $\mathcal{A}$ avoids $\mathcal{B}$ and $\mathcal{A}$ and $\mathcal{B}$ can be separated by a line. Let $\mathcal{P} = \mathcal{A} \cup \mathcal{B}$. Then the dual arrangement $\mathcal{P}^*$ contains a spoke path of length $k+2$, if k is even, or $k+3$, if k is odd.*

For a full proof we again refer to the full version. But we will briefly sketch the main steps of the construction.

Step 1: Let $\mathcal{A}^*$ and $\mathcal{B}^*$ denote the duals of $\mathcal{A}$ and $\mathcal{B}$, respectively. Draw $\mathcal{B}^*$ as a wiring diagram in color red. As $\mathcal{A}$ avoids $\mathcal{B}$ and $\mathcal{A}$ and $\mathcal{B}$ can be separated by a line, all lines of $\mathcal{A}^*$ cross the lines of $\mathcal{B}^*$ in the same order, so we can draw $\mathcal{A}^*$ as pseudolines that are straight and vertical in the region where they cross the red pseudolines and get a pseudoline arrangement that is isomorphic to $\mathcal{P}^*$. We call such a drawing an *extended diagram*. See Figure 3 for an example of an extended diagram. Let r_1 be the bottommost pseudoline at left infinity of the wiring diagram of $\mathcal{B}^*$. For some directed pseudoline g, we define the *color sequence* $c(g)$ of g by moving along g and writing for each crossing with another pseudoline an r or a b if the crossed pseudoline is red or blue, respectively. In particular, $c(r_1)$ denotes the color sequence defined by moving along r_1 from left infinity to right infinity.

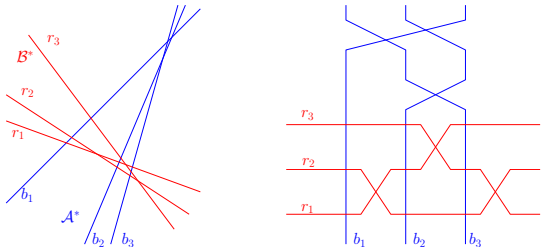


Figure 3: A line arrangement and its extended diagram.

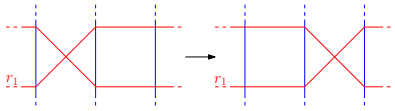


Figure 4: A right single crossing move.

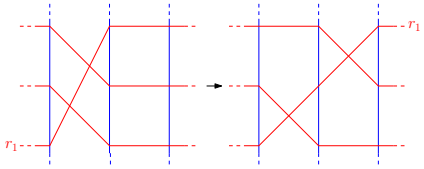


Figure 5: A right split crossing move.

Step 2: We modify the extended diagram using a sequence of moves. We use two different types of moves. For an illustration of the moves, see Figures 4 and 5. The *right (left) single crossing move* can be used if $c(r_1) = \ldots brbb \ldots$ ($c(r_1) = \ldots bbrb \ldots$). We move the crossing with the red pseudoline to the right (left), changing the color sequence of r_1 to $c(r_1) = \ldots bbrb \ldots$ ($c(r_1) = \ldots brbb \ldots$). The *right (left) split crossing move* can be used if there is more than one crossing with red pseudolines between two blue pseudolines, i.e., if $c(r_1) = \ldots brr \ldots rrbb \ldots$ ($c(r_1) = \ldots bbrr \ldots rrb \ldots$). We split the last of these crossings off and move it to the right (left), changing the color sequence of r_1 to $c(r_1) = \ldots brr \ldots rbrb \ldots$ ($c(r_1) = \ldots brbr \ldots rrb \ldots$). The same moves can also be defined if $c(r_1)$ starts with rbb or $r \ldots rbb$ (ends with bbr or $bbr \ldots r$). We do these moves until we reach a goal diagram in which r_1 has the color sequence $c(r_1) = brbrbr \ldots brb$ (note that $c(r_1)$ has length $2k - 1$). As in a split crossing move we split two consecutive r's and no move joins two r's, we can conclude that among the moves we need to reach the goal diagram, at most $k - 2$ are split crossing moves. The goal diagram is of course not isomorphic to $\mathcal{P}^*$ anymore.

Step 3: We draw two new directed pseudolines g_1 and g_2 in the goal drawing, representing cell paths given by the cells they intersect. Let C_0 be the unbounded cell that is under all red pseudolines and left of all blue pseudolines. Both g_1 and g_2 start in C_0

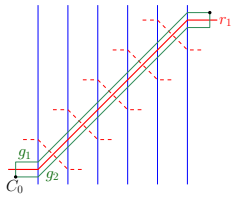


Figure 6: The goal diagram with the pseudolines g_1 and g_2.

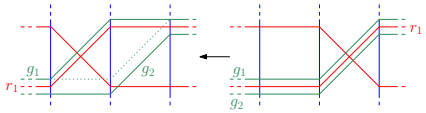


Figure 7: Reversing a single crossing move.

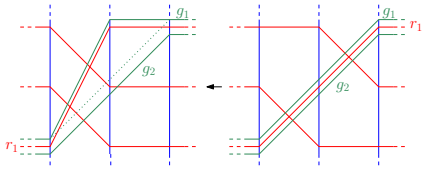


Figure 8: Reversing a split crossing move.

and end in the antipodal cell of C_0, but g_1 crosses r_1 first and then always stays at a small distance to it, whereas g_2 always stays at small distance to r_1 and crosses it at the very end. Then g_1 and g_2 have the color sequences $c(g_1) = rbrbrbr \ldots brb$ and $c(g_2) = brbrbr \ldots brbr$, see Figure 6 for an illustration. For any color sequence we call a subsequence $x_1, \ldots, x_j$ *semialternating* if j is even, i.e., $j = 2m$, and for every $i \leq m$ we have that $x_{2i-1} = r \Leftrightarrow x_{2i} = b$. By $\phi(g_1)$ and $\phi(g_2)$ we denote the length of the longest semialternating subsequences of $c(g_1)$ and $c(g_2)$, respectively. Note that by our construction of g_1 and g_2 we have that $\phi(g_1) = \phi(g_2) = 2k$.

Step 4: We reverse the moves to get back to our initial extended diagram. While doing so, we change g_1 and g_2 only if one of them crosses r_1 more than once. In that case we just delete the part between the newly introduced crossings and replace it with a pseudoline segment that stays at a small distance to r_1. For an illustration see Figures 7 and 8. In each step we only need to change either g_1 or g_2, but never both. Also, $\phi(g_1)$ or $\phi(g_2)$ only changes when we reverse a split crossing move, where it decreases by 2 only for the pseudoline that was modified.

Step 5: We reach the initial extended diagram, but with two additional directed pseudolines g_1 and g_2, representing cell paths. For both of these directed pseudolines, the longest semialternating subsequence of the color sequence represents a line-monotone AB-semialternating cell-path, i.e., a spoke path of length $\phi(g_1)$ or $\phi(g_2)$, respectively. In the goal diagram we

had $\phi(g_1) + \phi(g_2) = 4k$. While reversing the moves, this sum has only changed by the term -2 when we reversed a split crossing move. As we used at most $k-2$ split crossing moves, for the initial diagram we have $\phi(g_1) + \phi(g_2) \geq 4k - (k-2) \cdot 2 = 2k + 4$. The result now follows from the pigeonhole principle and the fact that by definition $\phi(g)$ is always even.

Corollary 3 *If a point set $\mathcal{P}$ contains two subsets $\mathcal{A}$ and $\mathcal{B}$ of size k, such that $\mathcal{A}$ avoids $\mathcal{B}$ and $\mathcal{A}$ and $\mathcal{B}$ can be separated by a line, then $\mathcal{P}$ contains a spoke set of size $\lceil \frac{k}{2} \rceil + 1$.*

Proof. Combine Lemma 1 and Lemma 2. □

Modifying the proof of Aronov et al. [1] for finding mutually avoiding sets in a point set, we can prove the following theorem:

Theorem 4 *Every point set of size n contains two point sets $\mathcal{A}$ and $\mathcal{B}$ of size $\lfloor \sqrt{\frac{n}{2} + 1} - 1 \rfloor$ such that $\mathcal{A}$ avoids $\mathcal{B}$ and $\mathcal{A}$ and $\mathcal{B}$ can be separated by a line.*

A proof of this can be found in the full version.

Corollary 5 *Every point set $\mathcal{P}$ of size n allows a spoke set of size at least $\sqrt{\frac{n}{8}}$.*

Proof. By Theorem 4, $\mathcal{P}$ contains two subsets $\mathcal{A}$ and $\mathcal{B}$ of size $\lfloor \sqrt{\frac{n}{2} + 1} - 1 \rfloor$ such that $\mathcal{A}$ avoids $\mathcal{B}$ and $\mathcal{A}$ and $\mathcal{B}$ can be separated by a line. Thus, by Corollary 3, the point set contains a spoke set of size

$$\left\lceil \frac{\lfloor \sqrt{\frac{n}{2} + 1} - 1 \rfloor}{2} \right\rceil + 1 \geq \left\lceil \sqrt{\frac{n}{8} + \frac{1}{4}} - 1 \right\rceil + 1 \geq \sqrt{\frac{n}{8}}.$$

□

It is worth mentioning that there are point sets that have no mutually avoiding subsets of size larger than $\mathcal{O}(\sqrt{n})$ [7]. However, it is not clear whether this still holds if we only insist that one of the subsets avoids the other one. So while there is no hope of finding larger crossing families by finding larger mutually avoiding subsets, it might still be possible to find larger spoke sets with this approach.

4 Spoke matchings

In this section we characterize a family of geometric matchings that arise from spoke sets. For this we need a few definitions:

Let e and f be two line segments and let s be the intersection of their supporting lines. If s lies in both e and f, we say that e and f *cross*. If s lies in f but not in e, we say that e *stabs* f and we call the vertex of e that is closer to s the *stabbing vertex* of e. If s lies neither in e nor in f, or if the supporting lines of e and f do not meet, we say that e and f are *parallel*.

A *stabbing chain* in a geometric matching are three edges, e, f and g, where e stabs f and f stabs g. We call f the *middle edge* of the stabbing chain.

Theorem 6 *A geometric matching M is a spoke matching if and only if it satisfies the following three conditions:*

(a) *no two edges are parallel,*

(b) *if an edge e stabs two other edges f and g, then the respective stabbing vertices of e lie inside the convex hull of f and g, and*

(c) *if there is a stabbing chain, then the stabbing vertex of the middle edge lies inside the convex hull of the other two edges.*

For a proof we refer to the full version. Note that the fact that every crossing family of size k induces a spoke set of size k can also be derived from this result, as it shows that every crossing family is a spoke matching. However, the family of spoke matchings also contains matchings that are not crossing families. In fact, it is even possible to construct a crossing-free spoke matching. In [5], it has been shown that there are sets of n points in general position that do not allow any matching satisfying conditions (a), (b) and (c) of size larger than $\frac{9}{20}n$. Hence we get the following corollary:

Corollary 7 *There are point sets of n points in general position that do not admit a spoke set of size larger than $\frac{9}{20}n$.*

References

[1] B. Aronov, P. Erdős, W. Goddard, D. J. Kleitman, M. Klugerman, J. Pach, and L. J. Schulman. Crossing families. In *Proceedings of the Seventh Annual Symposium on Computational Geometry, North Conway, NH, USA, June 10-12, 1991*, pages 351–356, 1991.

[2] P. Bose, F. Hurtado, E. Rivera-Campo, and D. R. Wood. Partitions of complete geometric graphs into plane trees. *Computational Geometry*, 34(2):116–125, 2006.

[3] R. Fulek and A. Suk. On disjoint crossing families in geometric graphs. *Electronic Notes in Discrete Mathematics*, 38:367–375, 2011.

[4] J. Pach and J. Solymosi. Halving lines and perfect cross-matchings. *Advances in Discrete and Computational Geometry*, 223:245–249, 1999.

[5] P. Schnider. Partitions and packings of complete geometric graphs with plane spanning double stars and paths. Master's thesis, ETH Zürich, 2015.

[6] P. Schnider. A generalization of crossing families. *CoRR*, abs/1702.07555, 2017.

[7] P. Valtr. On mutually avoiding sets. In *The mathematics of Paul Paul Erdős, II (R. L. Graham and J. Nešetřil, eds.) Algorithms and Combin. 14*, pages 324–332, 1997.

Ordered Level Planarity and Geodesic Planarity

Boris Klemz* Günter Rote*

Abstract

We introduce and study the problem Ordered Level Planarity which asks for a planar drawing of a graph such that vertices are placed at prescribed positions in the plane and such that every edge is realized as a y-monotone curve. This can be interpreted as a variant of Level Planarity in which the vertices on each level appear in a prescribed total order. We show $\mathcal{NP}$-completeness even for the special case that the number of vertices on each level is bounded by $\lambda = 2$ and that the maximum degree is $\Delta = 2$. This establishes a tight border of tractability since for $\lambda = 1$ the problem is in $\mathcal{P}$. Our result is motivated by the following applications.

We establish a connection to geodesic drawings. Geodesic Planarity asks for a planar drawing of a graph such that vertices are placed at prescribed positions in the plane and such that every edge e is realized as a polygonal path p composed of line segments with two adjacent directions from a given set S of directions symmetric with respect to the origin. Our results on Ordered Level Planarity imply $\mathcal{NP}$-hardness for any S with $|S| \geq 4$ even if the given graph is a matching. Katz, Krug, Rutter and Wolff claimed that for matchings Manhattan Geodesic Planarity is in $\mathcal{P}$ [GD'09]. Our results imply that this is incorrect unless $\mathcal{P} = \mathcal{NP}$. Further, our results imply the $\mathcal{NP}$-hardness of the Bi-Monotonicity problem.

We narrow the gap between tractability and $\mathcal{NP}$-hardness in the established hierarchy of Level Planarity variants. To this end, we provide reductions to T-Level Planarity, Clustered Level Planarity and Constrained Planarity. As a by-product, we strengthen previous $\mathcal{NP}$-hardness results. In particular, our reduction to Clustered Level Planarity generates instances with $\Delta = 2$, $\lambda = 2$ and only two non-trivial clusters.

1 Introduction

An *upward* planar drawing of a directed graph is a plane drawing where every edge $e = (u, v)$ is realized as a y-monotone curve that goes upward from u to v. Upward planar drawings provide a very natural way of visualizing a partial order on a set of items. The problem Upward Planarity of testing whether a directed graph has an upward planar drawing is $\mathcal{NP}$-complete [5]. However, if the y-coordinate of each vertex is prescribed, the problem can be solved in polynomial time [6]. Formally, this is captured by the notion of level graphs. A *level graph* $\mathcal{G} = (G, \gamma)$ is a directed graph $G = (V, E)$ together with a *level assignment* $\gamma : V \to \{0, \dots, h\}$ for G where γ is a surjective map with $\gamma(u) < \gamma(v)$ for every edge $(u, v) \in E$. Value h is the *height* of $\mathcal{G}$. The vertex set $V_i = \{v \mid \gamma(v) = i\}$ is called the i-th *level* of $\mathcal{G}$. Value $\lambda_i = |V_i|$ is the *width* of level i and the *level-width* λ of $\mathcal{G}$ is the maximum width of any level in $\mathcal{G}$. A *level* planar drawing of $\mathcal{G}$ is an upward planar drawing of G where the y-coordinate of each vertex v is $\gamma(v)$. The problem Level Planarity of testing whether a given level graph has a level planar drawing is solvable in linear time [6].

We introduce a natural variant of Level Planarity that takes into account a total order for the vertices on each level. An *ordered* level graph $\mathcal{G}$ is a triple $(G = (V, E), \gamma, \chi)$ where (G, γ) is a level graph and $\chi : V \to \{0, \dots, \lambda - 1\}$ is a *level ordering* for G. We require that χ restricted to domain V_i bijectively maps to $\{0, \dots, \lambda_i - 1\}$. An *ordered* level planar drawing of an ordered level graph $\mathcal{G}$ is a level planar drawing of (G, γ) where for every $v \in V$ the x-coordinate of v is $\chi(v)$. Thus, the position of every vertex is fixed. The problem Ordered Level Planarity asks whether a given ordered level graph has an ordered level planar drawing. We remark that in the above definitions, the x- and y-coordinates assigned via χ and γ merely act as a convenient way to encode total and partial orders respectively. In terms of realizability, the problems are equivalent to generalized versions were χ and γ map to the reals. In other words, the fixed vertex positions can be any points in the plane. All reductions and algorithms in this paper carry over to these generalized versions, if we pay the cost for presorting the vertices according to their coordinates.

We establish a connection between ordered level planar drawings and geodesic drawings. Let $S \subset \mathbb{R}^2$ be a finite set of directions symmetric with respect to the origin, i.e. for each direction $s \in S$, the reverse direction $-s$ is also contained in S. A plane drawing of a graph is *geodesic* with respect to S if every edge is realized as a polygonal path p composed of line segments with two adjacent directions from S.

*Institute of Computer Science, Freie Universität Berlin, Germany

Such path p is a geodesic with respect to the polygonal norm that corresponds to S. An instance of the decision problem Geodesic Planarity is a 4-tuple $\mathcal{G} = (G = (V, E), x, y, S)$ where G is a graph, x and y map from V to the reals and S is a set of directions as stated above. The task is to decide whether $\mathcal{G}$ has a *geodesic drawing*, that is, G has a geodesic drawing with respect to S in which every vertex $v \in V$ is placed at $(x(v), y(v))$.

Katz, Krug, Rutter and Wolff [7] study Manhattan Geodesic Planarity, which is the special case of Geodesic Planarity where set S consists of the two horizontal and the two vertical directions. Geodesic drawings with respect to this set of directions are also referred to as orthogeodesic drawings. Katz et al. [7] show that a variant of Manhattan Geodesic Planarity in which the drawings are restricted to the integer grid is $\mathcal{NP}$-hard even if G is a perfect matching. The proof is by reduction from 3-Partition and makes use of the fact the number of edges that can pass between two vertices on a grid line is bounded. In contrast, they claim that the standard version of Manhattan Geodesic Planarity is polynomial-time solvable for perfect matchings. To this end, they sketch a plane sweep algorithm that maintains a linear order among the edges that cross the sweep line. When a new edge is encountered it is inserted as low as possible subject to the constraints implied by the prescribed vertex positions. When asked for more details, the authors informed us that they are no longer convinced of the correctness of their approach. Unless $\mathcal{P} = \mathcal{NP}$, one of the results of our paper implies that their approach is indeed incorrect.

The results and layout of this abstract are as follows. In Section 3 we study the complexity of Ordered Level Planarity. While Upward Planarity is $\mathcal{NP}$-complete [5] in general but becomes polynomial-time solvable [6] for prescribed y-coordinates, we show that prescribing both x and y-coordinates renders the problem $\mathcal{NP}$-complete. Precisely, our results are summarized in the following theorem.

Theorem 1 Ordered Level Planarity *is $\mathcal{NP}$-complete, even for maximum degree $\Delta = 2$ and level-width $\lambda = 2$. For level-width $\lambda = 1$,* Ordered Level Planarity *can be solved in linear time.*

Theorem 1 states an explicit gap between tractability and $\mathcal{NP}$-hardness. We motivate this result with the multiple applications. In Section 2 we study the complexity of Geodesic Planarity. We utilize Theorem 1 to obtain the following:

Theorem 2 Geodesic Planarity *is $\mathcal{NP}$-hard for any set of directions S with $|S| \geq 4$ even for perfect matchings in general position.*

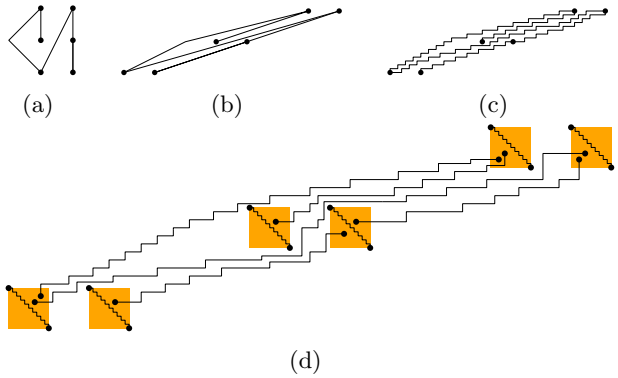


Figure 1: Reduction to Geodesic Planarity.

In the full version we provide reductions which establish Ordered Level Planarity as a special case of T-Level Planarity [1], Clustered Level Planarity [1] and Constrained Planarity [3]. As a by-product, we strengthen previous $\mathcal{NP}$-hardness results. In particular, we show that Clustered Level Planarity is $\mathcal{NP}$-hard even for instances with $\lambda = 2$, $\Delta = 2$ and only two non-trivial clusters. We observe that Geodesic Planarity restricted to instances as stated in Theorem 2 reduces immediately to Bi-Monotonicity [4] if S contains precisely the horizontal and vertical directions. Thus, we settle the latter problem's complexity.

2 Geodesic Planarity

In this section we sketch the proof of Theorem 2. To this end, we transform an Ordered Level Planarity instance $\mathcal{G}_o = (G_o = (V, E), \gamma, \chi)$ with maximum degree $\Delta = 2$ and level-width $\lambda = 2$ into a Geodesic Planarity instance $\mathcal{G}_g = (G_g, x, y, S)$ where G_g is a perfect matching. In this abstract we describe the reduction specifically for the case that the set S consists of precisely the horizontal and vertical directions. However, the construction is invariant under shearing and, thus, works for any prescribed set S of directions with $|S| \geq 4$. The reduction is carried out in two steps.

First, we transform $\mathcal{G}_o$ into a Geodesic Planarity instance $\mathcal{G}'_g = (G_o, x', \gamma, S)$ by translating the vertices of level V_i by $3i$ units to the right, see Fig.1a and Fig.1b. Clearly, every geodesic drawing of $\mathcal{G}'_g$ can be turned into an ordered level planar drawing of $\mathcal{G}_o$. On the other hand, consider an ordered level planar drawing of $\mathcal{G}_o$. W.l.o.g. all edges are realized as polygonal paths in which bend points occur only on the horizontal lines L_i through the levels V_i of $\mathcal{G}_o$, see Fig.1a. Further, assume that all bend points have an x-coordinate in the interval $[-1, 2]$. We translate all bend points on L_i by $3i$ units to the right, see Fig.1b. In the resulting drawing all edge-segments have a slope from interval $(0, \infty)$. Thus, since the

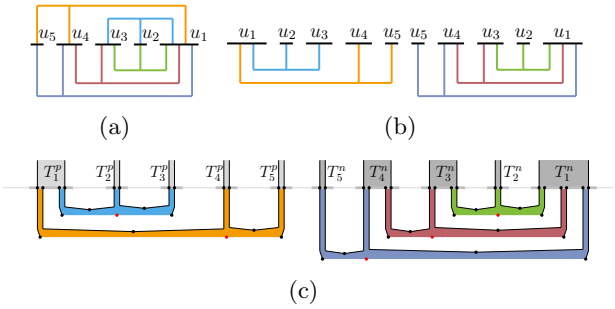


Figure 2: Representation of a planar monotone 3SAT formula and its usage in the reduction to Ordered Level Planarity.

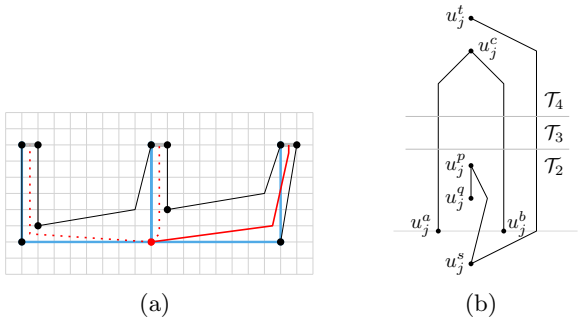


Figure 3: **(a)** The subgraph (black/red) created for every E-shape (blue) and the three respective gates (grey). **(b)** The variable gadget for u_j.

maximum degree is $\Delta = 2$ we can be replace all edge-segments by geodesic staircases that closely trace the segments, see Fig.1c.

In the second step we turn G_o into a perfect matching in order to obtain $\mathcal{G}_g$. To this end, we essentially split each vertex v by replacing it with a small gadget that fits inside a $1/4 \times 1/4$ square centered on v, see Fig.1d. The gadget contains a degree-1 vertex for every edge incident to v. In order to maintain equivalence we have to prevent non-incident edges from being drawn through the gadget square. To this end, we create a *blocking* edge between vertices in the top left and bottom right corners of the gadget square.

Note that all x-coordinates are distinct. Only the up to 8 vertices of the gadgets on each level may have duplicate y-coordinates. Thus, by placing these vertices more carefully we can guarantee that the assigned vertex positions are in *general position*, that is, no two vertices lie on a line with a direction from S.

3 Ordered Level Planarity

To obtain $\mathcal{NP}$-hardness of Ordered Level Planarity we perform a reduction from Planar Monotone 3-Satisfiability. In this $\mathcal{NP}$-hard [2] special case of 3SAT the input is a 3SAT formula φ together with a contact representation $\mathcal{R}$ of φ, see Fig. 2a. All variables are represented as line segments arranged on a line. Each clause c is represented as an E-shape that touches precisely the variables contained in c. Furthermore, all clauses are either *positive* or *negative*, i.e. they contain exclusively positive or negative literals, respectively. In $\mathcal{R}$ all negative clauses are below the line of variables and all positive clauses are above the line.

Recall that in terms of realizability Ordered Level Planarity is equivalent to the generalized version where γ and χ map to the reals. For the sake of convenience we will describe our construction in this generalized setting. We create an ordered level graph whose level assignment is partitioned into four tiers $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3, \mathcal{T}_4$. Each clause c_i is associated with a *clause edge* starting in $\mathcal{T}_1$ and ending in $\mathcal{T}_3$. In tier $\mathcal{T}_1$ we do the following. We take the top part of $\mathcal{R}$, rotate it by 180° and place it to the left of the bottom part, see Fig. 2b. For each E-shape we create a 11-vertex subgraph as illustrated in Fig. 3a. The red vertex is the bottom vertex of the clause edge belonging to the clause that corresponds to the E-shape. Observe that drawings of this subgraph are *unique* in the sense that the left-to-right order of vertices and edges intersected by a horizontal line through any of the vertices is the identical in every ordered level planar drawing of the subgraph. This is due to the fact that the order of the top 6 vertices is fixed since they are placed on the same level. As a consequence, the clause edge starting at the red vertex has to intersect one of the thick gray regions, which we call *gates*. Fig. 2c illustrates the entirety of $\mathcal{T}_1$. The subgraph induced by $\mathcal{T}_1$ has a unique drawing. Further, note that each gate is located in the line segment of one of the variables of $\mathcal{R}$. We bundle all gates that are located in the line segment of the same literal together by creating *tunnels* as depicted in the top of Fig. 2c. Observe that the clause edge of clause c_i has to be drawn inside the tunnel of one of the literals of c_i. This corresponds to the fact that in a satisfying truth assignment every clause has at least one satisfied literal.

We need to ensure that for each variable u_j either its positive tunnel T_j^p or negative tunnel T_j^n can be used, but not both. To this end, we create a *variable gadget* for each variable u_j, see Figure 3b. These gadgets start in $\mathcal{T}_2$ and end in $\mathcal{T}_4$. In $\mathcal{T}_2$ the gadget for u_j starts above all the gadgets of variables with smaller index. In $\mathcal{T}_4$ the gadget for u_j ends below all the gadgets of variables with smaller index. The tunnels T_j^p and T_j^n end inside the gadget of variable u_j on level $\gamma(u_j^q)$. We force all tunnels with index at least n to be drawn between u_j^a and u_j^b by subdividing the tunnel edges appropriately, see Fig. 4a. The *long* edge (u_j^s, u_j^t) has to be drawn left or right of u_j^c in $\mathcal{T}_4$. Accordingly it has to be drawn left of u_j^a or right of u_j^b in $\mathcal{T}_2$ and, thus, left or right of all the

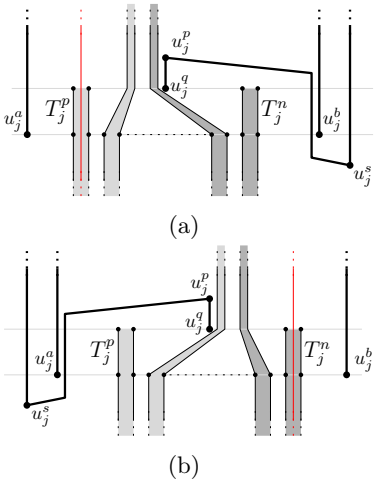


Figure 4: The two states of a variable gadet.

tunnels that are drawn between u_j^a and u_j^b. As a consequence, the *blocking* edge (u_j^s, u_j^p) is also drawn left (Fig. 4b) or right (Fig. 4a) of all tunnels. Together with the edge (u_j^q, u_j^p) it prevents clause edges from being drawn in T_j^p or T_j^n depending on whether the long edge and the blocking edge are drawn left or right.

We summarize the reduction. If there exists an ordered level planar drawing, then the clause edge of each clause is drawn inside of a tunnel that corresponds to one of its literals. Due to the variable gadgets, edges can only be drawn either inside the positive tunnel or inside the negative tunnel of a variable. Thus, we obtain a satisfying truth assignment. On the other hand, given a satisfying truth assignment we can create a drawing by placing the long edges of the variable gadgets according to the assignment. Fig. 5 illustrates how variable gadgets can be nested and clause edges can be drawn.

The resulting Ordered Level Planarity instance has maximum degree $\Delta = 2$. The level-width λ is linear in the input size, however, it can be decreased to $\lambda = 2$ by replacing a level with width $\lambda_i > 2$ with $\lambda_i - 1$ levels containing exactly two vertices each. For more details, we refer to the full version. For $\lambda = 1$ Ordered Level Planarity is solvable in linear time since Level Planarity can be solved in linear time [6].

Acknowledgements: We thank the authors of [7] for providing us with unpublished information regarding their plane sweep approach for Manhattan Geodesic Planarity.

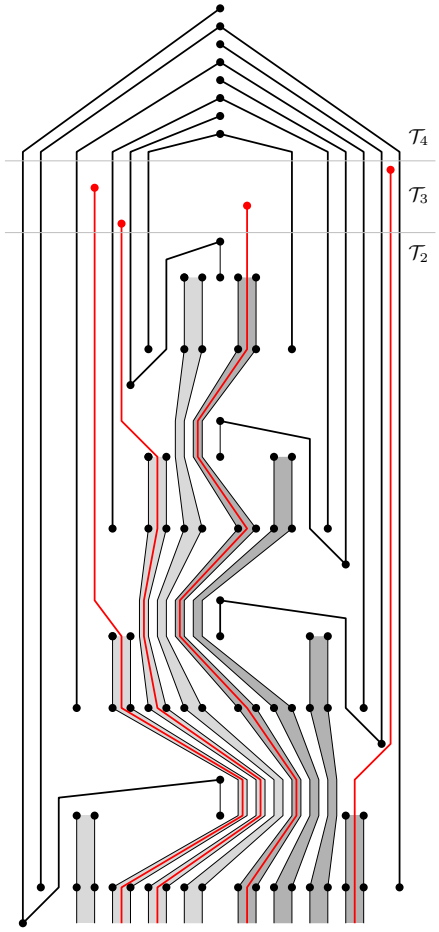


Figure 5: Nesting structure of the variable gadgets.

References

[1] Angelini, P., Da Lozzo, G., Di Battista, G., Frati, F., Roselli, V.: The importance of being proper: (in clustered-level planarity and t-level planarity). Theor. Comput. Sci. 571, 1–9 (2015)

[2] de Berg, M., Khosravi, A.: Optimal binary space partitions for segments in the plane. Int. J. Comput. Geometry Appl. 22(3), 187–206 (2012)

[3] Brückner, G., Rutter, I.: Partial and constrained level planarity. In: Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19. pp. 2000–2011 (2017)

[4] Fulek, R., Pelsmajer, M.J., Schaefer, M., Štefankovič, D.: Hanani–tutte, monotone drawings, and level-planarity. In: Thirty Essays on Geometric Graph Theory, pp. 263–287. Springer (2013)

[5] Garg, A., Tamassia, R.: On the computational complexity of upward and rectilinear planarity testing. SIAM J. Comput. 31(2), 601–625 (2001)

[6] Jünger, M., Leipert, S., Mutzel, P.: Level planarity testing in linear time. In: Graph Drawing, 6th International Symposium, GD'98, Montréal, Canada, August 1998, Proceedings. pp. 224–237 (1998)

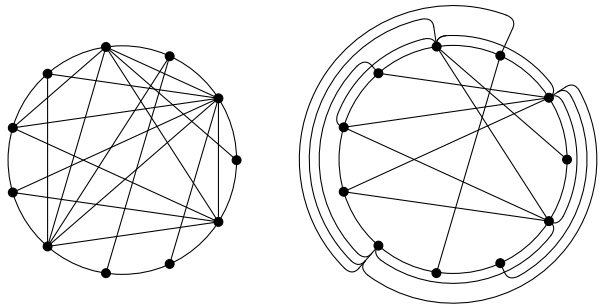
[7] Katz, B., Krug, M., Rutter, I., Wolff, A.: Manhattan-geodesic embedding of planar graphs. In: Graph Drawing, 17th International Symposium, GD 2009, Chicago, IL, USA, September 22-25, 2009. Revised Papers. pp. 207–218 (2009)

Minimizing crossings in constrained two-sided circular graph layouts

Fabian Klute* Martin Nöllenburg*

Abstract

Circular layouts are a popular graph drawing style, where vertices are placed on a circle and edges are drawn as straight chords. One way to reduce clutter caused by edge crossings is to use *two-sided circular layouts*, in which some edges are drawn as curves in the exterior of the circle. We study the problem of minimizing the crossings for a fixed cyclic vertex order by computing an optimal 1-plane set of exteriorly drawn edges. This relates to finding maximum-weight degree-constrained induced subgraphs in circle or overlap graphs.



(a) One-sided layout (b) Two-sided layout

Figure 1: Circular graph layouts

1 Introduction

Circular graph layouts are a popular drawing style to visualize graphs, which focuses on a clear positioning of the vertices on a circle, while the edges are drawn as straight-line chords of said circle. As it is often the case in graph drawing the crossings between the edges play a big role in optimizing the readability of the visualization. If the edges are drawn as chords all crossings are determined by the order of the vertices. Finding a vertex order that minimizes the crossings is NP-hard [4]. Heuristics and approximation algorithms have been studied in numerous papers, see e.g. [1].

Gansner and Koren [2] presented an approach to compute improved circular layouts for a given input graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ in a three-step process. The first step computes a vertex order that aims to minimize the overall edge length of the drawing, the second step determines a crossing-free subset of edges that are drawn outside the circle to reduce edge crossings in the interior, and the third step introduces edge bundling to save ink and reduce clutter in the interior.

Inspired by their approach we take a closer look at the second step of the above process, which, in other words, determines an outerplane subgraph to be drawn outside the circle such that the remaining crossings of the chords are minimized. Gansner and Koren [2] solve this problem in $O(|\mathcal{V}|^3)$ time[1]. In fact, the problem is equivalent to finding a maximum independent set in the corresponding circle graph $G = (V, E)$ (see Section 2). This graph has a vertex for each edge of $\mathcal{G}$ and an edge between each pair of crossing chords in the circular layout of $\mathcal{G}$. The maximum independent set problem in a circle graph can be solved in $O(\ell)$ time [6], where $\ell \in \Omega(|\mathcal{E}|) \cap O(|\mathcal{E}|^2)$ is the total chord length in an interval representation of G.

We generalize the problem from outerplane graphs to outer k-plane graphs, i.e., we ask for an edge set to be drawn outside the circle such that none of these edges has more than k crossings. For $k = 0$ this is the same problem considered by Gansner and Koren [2]. In this paper we present an efficient algorithm based on dynamic programming for the case $k = 1$, where at most one crossing per exterior edge is permitted. Of course, this is only a first step towards solving the general case. Yet, due to non-local dependencies that occur for $k \geq 2$, we do not see an obvious way of extending our algorithm.

2 Optimizing interior crossings

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph and π a cyclic order of $\mathcal{V}$. We arrange the vertices in this order on a circle C and draw edges as straight chords to obtain a (one-sided) *circular drawing* Γ, see Fig. 1a. Note that the number of crossings of Γ is fully determined by π. Our goal in this paper is to draw a subset of edges on the exterior side of C in order to reduce the number of edge crossings.

In a *two-sided circular drawing* Δ of $\mathcal{G}$ and π we still draw all vertices on a circle C according to π, but we split the edges into two disjoint sets $\mathcal{E}_1$ and $\mathcal{E}_2$ with $\mathcal{E}_1 \cup \mathcal{E}_2 = \mathcal{E}$. The edges in $\mathcal{E}_1$ are drawn as straight chords, while the edges in $\mathcal{E}_2$ are drawn as curves in the exterior of C, see Fig. 1b. Rather than asking for a set $\mathcal{E}_2$ that globally minimizes the crossings in Δ, which is equivalent to the NP-hard fixed linear crossing

*Algorithms and Complexity Group, TU Wien, Austria

[1]The paper actually claims $O(|\mathcal{V}|^2)$ time without a proof; the immediate running time of their algorithm is $O(|\mathcal{V}|^3)$.

minimization problem in 2-page book embeddings [5], we add the additional constraint that the exterior drawing induced by $\mathcal{E}_2$ is *outer k-plane*, i.e., each edge in $\mathcal{E}_2$ is crossed by at most k other edges in $\mathcal{E}_2$. This is motivated by the fact that exterior edges are harder to read and should not be further impaired by too many crossings.

Instead of working with $\mathcal{G}$ and π directly we consider the corresponding *circle graph* $G = G_{\mathcal{G},\pi} = (V, E)$ of $\mathcal{G}$ and π, where V has one vertex for each edge in $\mathcal{E}$ and two vertices $u, v \in V$ are connected by an edge (u, v) in E if and only if the edges corresponding to u and v cross in the circular layout Γ. So the number of vertices $|V| = |\mathcal{E}|$ equals the number of edges of $\mathcal{G}$ and the number of edges $|E|$ is the number of crossings of Γ. We further assign to every vertex $v \in V$ a weight $w(v) \in \mathbb{R}^+$ and to every edge $(u, v) \in E$ a weight $w(u, v) \in \mathbb{R}^+$.

Finding the set $\mathcal{E}_2$ can now be modeled as a constrained maximum induced subgraph problem on the circle graph G. The general problem can be stated as follows.

Definition 1 *(*Max-Weight Deg-k Induced Subgraph*) Given a weighted graph $G = (V, E)$ and $k \in \mathbb{N}$ find a set $V' \subset V$ such that the induced subgraph $G[V'] = (V', E')$ has maximum degree k and maximizes the sum*

$$W = \sum_{v \in V'} w(v) - \sum_{(u,v) \in E'} w(u, v).$$

For general graphs it is NP-hard [7] to find such a subgraph, but restricting the graph class of G to circle graphs makes the problem significantly easier as we will show in Section 3.

It remains to model our constrained crossing minimization problem for two-sided circular layouts as an instance of Max-Weight Deg-k Induced Subgraph. We define the weights of G as $w(v) = \deg(v)$ for all $v \in V$ and as $w(u, v) = 1$ or, alternatively, as $w(u, v) = 2$ for all $(u, v) \in E$, depending on the type of crossings to minimize.

Lemma 1 *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph with cyclic vertex order π and $k \in \mathbb{N}$. Then a maximum-weight degree-k induced subgraph in G induces an outer k-plane graph in Γ that minimizes the number of crossings in the corresponding two-sided layout Δ.*

Proof. Let $V^* \subset V$ be a vertex set that induces a maximum-weight degree-k subgraph in $G = G_{\mathcal{G},\pi}$. Since vertices in G correspond to edges in $\mathcal{G}$, we can choose $\mathcal{E}^* = V^*$ as the set of exterior edges in Δ. Each edge in G corresponds to a crossing in the circular layout Γ. Hence each edge in $G[V^*]$ corresponds to an exterior crossing in Δ. Since the maximum degree of $G[V^*]$ is k, no exterior edge in Δ has more than k crossings.

The degree of a vertex $v \in V^*$ (and thus its weight $w(v)$) equals the number of crossings that are removed from Γ by drawing the corresponding edge in E' in the exterior part of Δ. However, if two vertices in V^* are connected by an edge, their corresponding edges in Δ cross in the exterior part of Δ and we need to add a correction term. For edge weights $w(u, v) = 1$ the weight W maximized by V^* equals the number of crossings that are removed from the interior part of Δ. For $w(u, v) = 2$, the weight W equals the number of crossings that are removed from the interior, but excluding those that are simply shifted to the exterior of Δ. □

3 Efficient Algorithm based on Overlap Graphs

In this section we use the known connection between circle graphs and overlap graphs, which are subgraphs of interval graphs, to design an efficient algorithm for our crossing minimization problem.

The key concept is to distinguish proper and non-proper overlaps of intervals. Let $\mathcal{I}$ be a set of intervals with distinct endpoints. For two intervals $I = [a, b], J = [c, d] \in \mathcal{I}$ we say that they overlap *properly* if either $a < c < b < d$ or $c < a < d < b$. We say that I *nests* J if $a < c < d < b$. Obviously, if two intervals intersect, they either overlap properly or one nests the other. For an interval $I \in \mathcal{I}$ we define the set $\mathcal{P}(I, \mathcal{I}) = \{J \mid J \in \mathcal{I} \text{ and } I \text{ properly overlaps } J\}$. By $\mathcal{P} = \cup_{I \in \mathcal{I}} \{(I, J) \mid J \in \mathcal{P}(I, \mathcal{I})\}$ we denote the set of all properly overlapping interval pairs of $\mathcal{I}$. Likewise we define $\mathcal{N}(I, \mathcal{I}) = \{J \mid J \in \mathcal{I} \text{ and } I \text{ nests } J\}$.

Given a set of intervals $\mathcal{I}$ we define the *overlap graph* of $\mathcal{I}$ as the graph $G_{\mathcal{I}} = (V, E)$ that has a vertex for each interval in $\mathcal{I}$ and an edge for each pair of properly overlapping intervals. In contrast to interval graphs, two nested intervals do not define an edge in the overlap graph.

Let ρ be the maximum degree of the overlap graph of $\mathcal{I}$ and let $\delta = \max\{|\mathcal{N}(I, \mathcal{I})| \mid I \in \mathcal{I}\}$ be the maximum number of intervals nested by any interval in $\mathcal{I}$. We define the parameter $\gamma = \max\{\rho, \delta\}$ as an upper bound on the number of intersections per interval.

As shown by Gavril [3] circle graphs and overlap graphs are isomorphic. The idea is to cut the circle C between two arbitrary vertices and project the chords onto the real line below C. Each chord is then represented by an interval and two chords intersect if and only if their projected intervals overlap properly, see Figure 2.

We rephrase Definition 1 in terms of interval representations of circle graphs. The weights can be taken directly from the circle graph $G_{\mathcal{G},\pi} = (V, E)$. For each interval $I \in \mathcal{I}$ corresponding to $v \in V$ we set $w(I) = w(v)$ and for each pair of properly overlapping intervals $(I, J) \in \mathcal{P}(\mathcal{I})$ corresponding to edge $e \in E$ we set $w(I, J) = w(e)$.

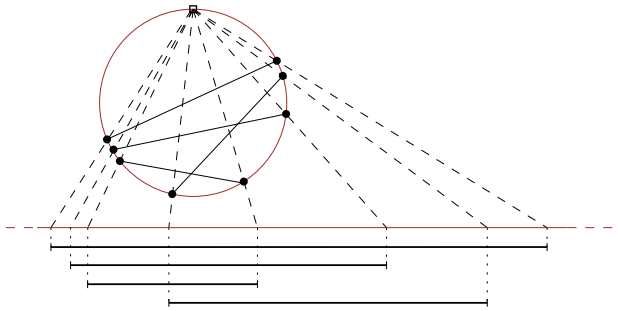


Figure 2: An example projection of a circle graph to a set of intervals with an isomorphic overlap graph.

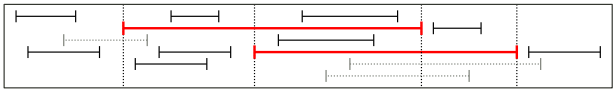


Figure 3: Split along the two red intervals. The dotted intervals are discarded and we recurse on the five sets with black intervals.

Definition 2 *(Max-Weight k-Intersection Set) Given an interval set $\mathcal{I}$ find a subset $\mathcal{I}' \subseteq \mathcal{I}$ such that no interval $I \in \mathcal{I}'$ has more than k proper intersections with other intervals in $\mathcal{I}'$ and the sum*

$$W = \sum_{I \in \mathcal{I}'} w(I) - \sum_{(I,J) \in P(\mathcal{I}')} w(I,J)$$

is maximized.

Since circle graphs and overlap graphs are isomorphic, we can also solve Max-Weight k-Intersection Set in order to solve our crossing minimization problem for two-sided circular layouts. In this paper, we restrict our attention to the case $k = 1$, i.e., finding an outer 1-plane edge set E' or, equivalently, finding an interval set with at most one proper intersection per interval.

3.1 Properties of max-weight 1-intersection sets

Before we describe our algorithm in Section 3.2 we introduce some notation and properties for splitting an interval set into subsets. Let $\mathcal{I}$ be a set of intervals. We say $\mathcal{I}$ has *common point* $x \in \mathbb{R}$ if $x \in I$ for all intervals $I \in \mathcal{I}$. For a general set of intervals $\mathcal{I}$ we define $\mathcal{I}|x = \{I \in \mathcal{I} \mid x \in I\}$ as the set of all intervals in $\mathcal{I}$ with common point x.

Further, for $x, y \in \mathbb{R} \cup \{\pm\infty\}$ with $x \leq y$ we define the set $\mathcal{I}[x,y] = \{I \in \mathcal{I} \mid I \subseteq [x,y]\}$. For any $x \leq y$ an interval set $\mathcal{I}[x,y]$ can be *split along* an interval $I = [a,b] \in \mathcal{I}$ into the three sets $\mathcal{I}[x,a]$, $\mathcal{I}[a,b]$, $\mathcal{I}[b,y]$. All intervals which are not contained in one of the three sets are discarded.

Finally we can split any $\mathcal{I}[x,y]$ along a pair of proper intersecting intervals $I = [a,b], J = [c,d] \in \mathcal{I}$. Without loss of generality let $a < c < b < d$. Then the split creates the five sets $\mathcal{I}[x,a]$, $\mathcal{I}[a,c]$, $\mathcal{I}[c,b]$, $\mathcal{I}[b,d]$, $\mathcal{I}[d,y]$. Again, all intervals which are not contained in one of the five sets are discarded. An example is shown in Figure 3.

Lemma 2 *Let $\mathcal{I}$ be a set of intervals. For any $x \in \mathbb{R}$ at most two properly intersecting intervals $I, J \in \mathcal{I}|x$ can be part of a max-weight 1-intersection set on $\mathcal{I}$.*

Proof. Assume there is a third interval $K \in \mathcal{I}|x$ in a max-weight 1-intersection set, which properly overlaps I or J or both. This K cannot be added to the solution set without creating at least one interval with more than one intersection, which is not allowed by definition. □

For an interval set $\mathcal{I}$ we call $I = [a,b] \in \mathcal{I}$ the *left-most* interval, if $a < a'$ for all $[a',b'] \in \mathcal{I} \setminus I$. We define the *left interval set* as $\mathscr{L}(\mathcal{I}) = P(I,\mathcal{I}) \cup N(I,\mathcal{I})$, the set of intervals intersecting the left-most interval I.

Lemma 3 *Let $\mathcal{I}$ be an interval set, $\mathcal{I}_o$ a max-weight 1-intersection set of $\mathcal{I}$ and $I \in \mathcal{I}$ the left-most interval. Then either $I \in \mathcal{I}_o$ or there exists at least one interval $J \in \mathcal{I}_o$ such that $J \in \mathscr{L}(\mathcal{I})$.*

Proof. Let $\mathcal{I}'_o$ be a max-weight 1-intersection set of $\mathcal{I}$ such that neither I nor an interval $J \in \mathscr{L}(\mathcal{I})$ is part of $\mathcal{I}'_o$. That is, there is no interval $K \in \mathcal{I}'_o$ that properly intersects I or is nested by I, but then $\mathcal{I}'_o \cup \{I\}$ is a solution to the max-weight 1-intersection set problem on $\mathcal{I}$ with larger weight which contradicts the optimality of $\mathcal{I}'_o$. □

3.2 Algorithm for the max-weight 1-intersection set problem

We use a dynamic programming algorithm to solve the max-weight 1-intersection set problem. The principal idea is to split a set of intervals $\mathcal{I}$ in each step along one interval or two properly intersecting intervals into smaller independent subsets. By Lemma 3 we do not have to consider splits along arbitrary intervals, but can choose either single intervals from $\mathscr{L}(\mathcal{I})$ or pairs of properly intersecting intervals, where at least one of them is in $\mathscr{L}(\mathcal{I})$.

We define a two-dimensional table T, in which we store the weight of an optimal local solution for each subinstance $\mathcal{I}[x,y]$ as the entry $T[x,y]$. Since for all relevant splits x and y are start- or end-points of intervals in $\mathcal{I}$ this table has size quadratic in the number of intervals. The best global solution corresponds to entry $T[-\infty,\infty]$, where $\pm\infty$ are symbolic dummy coordinates.

Picking one Interval A single interval $I = [a,b] \in \mathscr{L}(\mathcal{I}[x,y])$ is chosen as a candidate for the optimal solution. This gives us a split along one interval and three subinstances to consider. The optimal solution

$T_1[x,y]$ between x and y, when splitting along one interval I, is the maximum across these splits plus the weight of the interval I.

$$T_1[x,y] = \max_{I \in \mathscr{L}(\mathcal{I}[x,y])} \big\{T[x,a]+T[a,b]+T[b,y]+w(I)\big\}.$$

Since we consider every interval in $\mathscr{L}(\mathcal{I}[x,y])$ this step maximizes over $O(\gamma)$ sub-cases, one for each interval in $\mathscr{L}(\mathcal{I}[x,y])$ which has size at most $\gamma = \max\{\delta,\rho\}$.

Picking two Intervals Two properly intersecting intervals $I = [a,b] \in \mathscr{L}(\mathcal{I}[x,y])$ and $J = [c,d] \in \mathcal{P}(I,\mathcal{I}[x,y])$ are chosen as candidates for the optimal solution. This gives us a split along two intervals and five subinstances to consider. The optimal solution $T_2[x,y]$ for the set $\mathcal{I}[x,y]$ is the maximum across the possible splits generated by pairs of properly intersecting intervals. The weight of an individual split is the weight of the optimal solutions of the generated subinstances plus the weight of the two chosen intervals I, J minus the weight attributed to the pair (I,J).

$$T_2[x,y] = \max_{\substack{I \in \mathscr{L}(\mathcal{I}[x,y]) \\ J \in \mathcal{P}(I,\mathcal{I}[x,y])}} \big\{T[x,a] + T[a,c] + T[c,b] + T[b,d] + T[d,y] + w(I) + w(J) - w(I,J)\big\}.$$

Since we consider every pair of properly intersecting intervals, one of which in $\mathscr{L}(\mathcal{I}[x,y])$, this case maximizes over $O(\gamma^2)$ sub-cases.

Maximizing over both possibilities for the split we obtain the optimal local solution $T[x,y]$ as

$$T[x,y] = \max\{T_1[x,y], T_2[x,y]\}. \tag{1}$$

The set of intervals forming an optimal solution of the max-weight 1-intersection set problem can be recovered using the standard process of backtracking the decisions made by the maximization steps.

Theorem 4 *The* Max-Weight 1-Intersection Set *problem for a set of intervals $\mathcal{I}$ can be solved in $O(\gamma^2 n^2)$ time, where $n = |\mathcal{I}|$ and γ is an upper bound on the number of intersections per interval.*

Proof. The time to compute an entry $T[x,y]$ is dominated by the case of splitting along a pair of intervals, which requires $O(\gamma^2)$ time as argued above. Since T has size $O(n^2)$, the total computation time is $O(\gamma^2 n^2)$.

It remains to show the correctness. Let $\mathcal{I}_o$ be an optimal solution to the max-weight 1-intersection set problem on $\mathcal{I}$. By definition $\mathcal{I}_o$ can be decomposed into pairs $I, J \in \mathcal{I}_o$ such that I and J intersect properly, and single intervals $K \in \mathcal{I}_o$ such that no other interval in $\mathcal{I}_o$ overlaps K properly.

The proof is by induction over the number of intervals in $\mathcal{I}_o$. In case $\mathcal{I}_o$ consists of a single interval or two properly intersecting intervals these have to be in $\mathscr{L}(\mathcal{I})$ by Lemma 3. Our algorithm considers exactly all single intervals and pairs of properly intersecting intervals of $\mathscr{L}(\mathcal{I})$ in its first step, in particular it consider the intervals in $\mathcal{I}_o$.

So let $\mathcal{I}_o$ be an optimal solution with more than two intervals or two single intervals. By Lemma 3, $\mathcal{I}_o$ must contain a single interval from $\mathscr{L}(\mathcal{I})$ or a pair with one interval from $\mathscr{L}(\mathcal{I})$ along which we can split $\mathcal{I}_o$. With this split we create either three or five smaller and independent subinstances. For each subinstance we can compute an optimal solution by induction hypothesis. Since our algorithm also considers that particular split, our solution is at least as good as $\mathcal{I}_o$. In the end we return exactly $\mathcal{I}_o$ or a solution with equal weight. □

Combining the results from above we can conclude with the following result on two-sided layouts.

Corollary 1 *Given a graph $\mathcal{G} = (\mathcal{V},\mathcal{E})$ and a cyclic vertex order π a crossing-minimal two-sided drawing Δ with outer 1-plane exterior edge set can be computed in $O(\gamma^2|\mathcal{E}|^2)$ time, where γ is the thickness of the overlap graph derived from the circle graph $G_{\mathcal{G},\pi}$.*

References

[1] M. Baur and U. Brandes. Crossing reduction in circular layouts. In *Graph-Theoretic Concepts in Computer Science (WG'04)*, volume 3353 of *LNCS*, pages 332–343. Springer Berlin Heidelberg, 2004.

[2] E. R. Gansner and Y. Koren. Improved circular layouts. In *Graph Drawing (GD'06)*, volume 4372 of *LNCS*, pages 386–398. Springer, 2007.

[3] F. Gavril. Algorithms for a maximum clique and a maximum independent set of a circle graph. *Networks*, 3(3):261–273, 1973.

[4] S. Masuda, T. Kashiwabara, K. Nakajima, and T. Fujisawa. On the NP-completeness of a computer network layout problem. In *Circuits and Systems (ISCAS'87)*, pages 292–295. IEEE, 1987.

[5] S. Masuda, K. Nakajima, T. Kashiwabara, and T. Fujisawa. Crossing minimization in linear embeddings of graphs. *IEEE Trans. Computers*, 39(1):124–127, 1990.

[6] G. Valiente. A new simple algorithm for the maximum-weight independent set problem on circle graphs. In *Algorithms and Computation (ISAAC'03)*, volume 2906 of *LNCS*, pages 129–137. Springer, 2003.

[7] M. Yannakakis. Node-and edge-deletion NP-complete problems. In *Theory of Computing (STOC'78)*, pages 253–264. ACM, 1978.

Irrational Guards are Sometimes Needed*

Mikkel Abrahamsen Anna Adamaszek† Tillmann Miltzow‡

Abstract

In this paper we study the *art gallery problem*, which is one of the fundamental problems in computational geometry. The objective is to place a minimum number of guards inside a simple polygon such that the guards together can see the whole polygon. We say that a guard at position x sees a point y if the line segment xy is fully contained in the polygon.

Despite an extensive study of the art gallery problem, it remained an open question whether there are polygons given by integer coordinates that require guard positions with irrational coordinates in any optimal solution. We give a positive answer to this question by constructing a *monotone* polygon with integer coordinates that can be guarded by three guards only when we allow to place the guards at points with irrational coordinates. Otherwise, four guards are needed.

1 Introduction

For a polygon $\mathcal{P}$ and points $x, y \in \mathcal{P}$, we say that x *sees* y if the interval xy is contained in $\mathcal{P}$. A *guard set* S is a set of points in $\mathcal{P}$ such that every point in $\mathcal{P}$ is seen by some point in S. The points in S are called *guards*. The *art gallery problem* is to find a minimum cardinality guard set for a simple polygon $\mathcal{P}$ on n vertices. The polygon $\mathcal{P}$ is considered to be filled, i.e., it consists of a closed polygonal curve in the plane and the bounded region enclosed by this curve.

This classical version of the art gallery problem has been originally formulated in 1973 by Victor Klee (see the book of O'Rourke [15, page 2]). It is often referred to as the *interior-guard art gallery problem* or the *point-guard art gallery problem*, to distinguish it from other versions that have been introduced over the years.

In 1978, Steve Fisk provided an elegant proof that $\lfloor n/3 \rfloor$ guards are always sufficient and sometimes necessary to guard a polygon with n vertices [11]. Five years earlier, Victor Klee had posed this question to Václav Chvátal, who soon gave a more complicated solution [6]. Since then, the art gallery problem has been extensively studied, both from the combinatorial and the algorithmic perspective. Most of this research, however, is not focused directly on the classical art gallery problem, but on its numerous versions, including different definitions of visibility, restricted classes of polygons, different shapes of the guards (point/line segment), restrictions on the positions of the guards, etc. For more detailed information we refer the reader to the following surveys [15, 19, 21].

Despite extensive research on the art gallery problem, no combinatorial algorithm for finding an optimal solution is known. The only exact algorithm is attributed to Micha Sharir (see [8]), who has shown that in $n^{O(k)}$ time one can find a guard set consisting of k guards, if such a guard set exists. This result is obtained by using standard tools from real algebraic geometry [2], and it is not known how to find an optimal solution without using this powerful machinery (see [3] for an analysis of the very restricted case of $k = 2$). To stress this even more: Without the tools from algebraic geometry, we would not know if it is decidable whether a guard set of size k exists or not! Some recent lower bounds [4] based on the exponential time hypothesis suggest that there might be no better exact algorithms than the one by Sharir.

To explain the difficulty in constructing exact algorithms, we want to emphasize that it is *not* known whether the decision version of the art gallery problem (i.e., the problem of deciding whether there is a guard set consisting of k guards, where k is a parameter) lies in the complexity class NP. While NP-hardness and APX-hardness of the art gallery problem have been shown for different classes of polygons [5, 9, 13, 14, 16, 18, 20], the question of whether the point-guard art gallery problem is in NP remains open. A simple way to show NP-membership would be to prove that there always exists an optimal set of guards with *rational* coordinates of polynomially bounded description.

Indeed, Sándor Fekete posed at MIT in 2010 and at Dagstuhl in 2011 an open problem, asking whether there are polygons requiring irrational coordinates in an optimal guard set [10]. The question has been

*Mikkel Abrahamsen is partially supported by Mikkel Thorup's Advanced Grant from the Danish Council for Independent Research under the Sapere Aude research career programme. Anna Adamszek is supported by the Danish Council for Independent Research DFF-MOBILEX mobility grant. Tillmann Miltzow is supported by the ERC grant "PARAMTIGHT: Parameterized complexity and the search for tight complexity results", no. 280152.

†University of Copenhagen, Denmark. {miab,anad}@di.ku.dk

‡Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI), t.miltzow@gmail.com

raised again by Günter Rote at EuroCG 2011 [17]. It has also been mentioned by Rezende *et al.* [7]: "it remains an open question whether there are polygons given by rational coordinates that require optimal guard positions with irrational coordinates". A similar question has been raised by Friedrichs *et al.* [12]: "[...] it is a long-standing open problem for the more general Art Gallery Problem (AGP): For the AGP it is not known whether the coordinates of an optimal guard cover can be represented with a polynomial number of bits".

Our results. We answer the open question of Sándor Fekete. Recall that a polygon $\mathcal{P}$ is called *monotone* if there exists a line l such that for every line l' orthogonal to l it holds that $\mathcal{P} \cap l'$ is an interval.

Theorem 1 *There is a simple monotone polygon $\mathcal{P}$ with integer coordinates of the vertices such that*

(i) $\mathcal{P}$ can be guarded by 3 guards placed at points with irrational coordinates, and

(ii) an optimal guard set of $\mathcal{P}$ with guards at points with rational coordinates has size 4.

Other related work. A new line of research focuses on implementing algorithms that are capable of solving instances of the art gallery problem with thousands of vertices, giving a solution which is close to the optimal one, see the recent survey by Rezende *et al.* [7]. They explain that many practical algorithms rely on "routines to find candidates for discrete guard and witness locations." We show that this technique inevitably leads to sub-optimal solutions unless irrational candidate locations are also considered. We believe that our example and techniques are a good starting point to construct benchmark instances for implementations of art gallery algorithms. Benchmark instances serve to validate the quality of algorithms. Using the same instances when comparing different algorithms makes the results comparable.

A problem related to the art-gallery problem is the *terrain guarding problem*. In this problem, an x-monotone polygonal curve c (i.e., the terrain) is given. The region R above the curve c has to be guarded, and the guards are restricted to lie on c. Similarly as in our problem, a guard x sees a point y if xy is contained in the region R. Although the solution space of the terrain guarding instance is the continuous polygonal curve c, a discretization of the solution space has been recently described by Friedrichs *et al.* [12]. Given a terrain with n vertices at integer position, they describe a set $S \subset \mathbb{Q}$ of size $O(n^3)$, computable in polynomial time, such that there is an optimal guard placement restricted to S. It follows that for the terrain guarding problem the phenomenon with irrational numbers does not appear, and also the decision version of the terrain guarding problem is in NP.

2 Intuition

In this section, we explain the key ideas behind the construction of the polygon $\mathcal{P}$. Our presentation is informal, see the full-version for a formal proof and complete description of the polygon. Here we omit all "scary" computations and focus on conveying the big picture.

Define a *rational point* to be a point with two rational coordinates. An *irrational point* is a point which is not rational. A *rational line* is a line that contains two rational points. An *irrational line* is a line that is not rational.

Forcing a Guard on a Line Segment. Consider the drawing of the polygon $\mathcal{P}$ in Figure 2. We will now explain an idea of how three pairs of triangular pockets can enforce three guards on three line segments within $\mathcal{P}$.

Consider the two triangular pockets in Figure 1. The blue line segment contains one edge of each of these pockets, and the interiors of the pockets are at different sides of the line segment. A guard which sees the point t must be placed within the orange triangular region, and guard which sees b must be placed within the yellow triangular region. Thus, a single guard can see both t and b only if it is on the blue line segment tb, which is the intersection of the two regions.

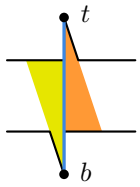


Figure 1: The only way that one guard can see both t and b is when the guard is on the blue line.

Consider now the case that we have k pairs of triangular pockets, and no two regions corresponding to different pairs of pockets intersect. In order to guard the polygon with k guards, there must be one guard on the line segment corresponding to each pair. Our polygon $\mathcal{P}$ has three such pairs of pockets (see Figure 2), and it can be checked that the corresponding

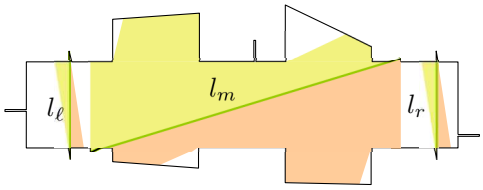


Figure 2: The only way to guard the polygon with three guards requires one guard on each of the green lines l_ℓ, l_m, l_r.

regions do not intersect.

Restricting a Guard to a Region Bounded by a Curve. For the following discussion, see the Figure 3 and notation therein. We want to guard the polygon from Figure 3 using two guards, g_1 and g_2. We assume that g_1 is forced to be on the blue vertical line l.

Consider some position of g_1 on l, such that g_1 can see at least one point of the top edge e_t of the top quadrilateral pocket, and at least one point of the bottom edge e_b of the bottom quadrilateral pocket. Let p_t and p_b denote the leftmost points seen by g_1 on e_t and e_b, respectively. Observe that p_t moves to the right if g_1 moves up, and to the left if g_1 moves down. The point p_b behaves in the opposite way when g_1 is moved. Consider some fixed position of g_1 on the blue line, and the corresponding positions of p_t and p_b. Let b be the bottom right corner of the top pocket, and d the top right corner of the bottom pocket. Let i be the intersection point of the line containing p_t and b, with the line containing p_b and d. The points b, d, i define a triangular region Δ. It is clear that if we place the guard g_2 anywhere inside Δ, then g_1 and g_2 will together see the entire polygon. On the other hand, if we place g_2 to the right of Δ, then g_1 and g_2 will not see the entire polygon, as some part of the top or the bottom pocket will not be seen.

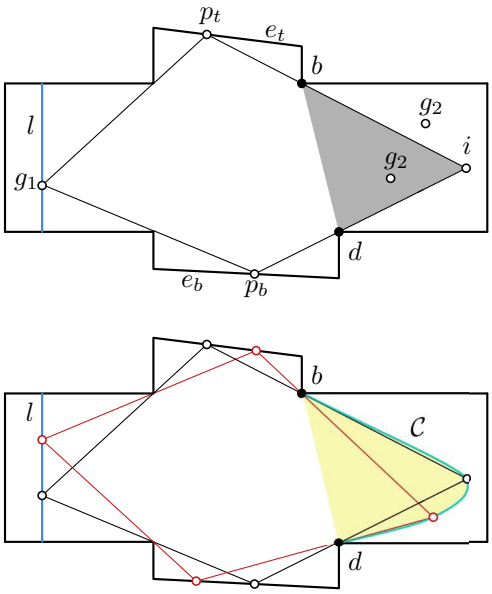


Figure 3: Top: The guard g_2 must be inside the shaded triangular region (or to the left of it) in order to guard the entire part of the polygon that is not seen by g_1. Bottom: All possible positions of the point i define a simple curve $\mathcal{C}$.

Now, let us move the guard g_1 along the blue line. Each position of g_1 yields some intersection point i. We denote the union of all these intersection points by $\mathcal{C}$ (see the bottom picture in Figure 3). It is easy to see that $\mathcal{C}$ is a simple curve. We can compute a parameterization of $\mathcal{C}$ since we have described how to construct the point i as a function of the position of g_1.

Note that g_2 sees a larger part of *both* pockets if it is moved horizontally to the left and a smaller part of *both* pockets if it is moved horizontally to the right. Consider a fixed position of g_2 on or to the right of the segment bd. Let g_2' be the horizontal projection of g_2 on $\mathcal{C}$. Let g_1 be the unique position on the blue line such that g_1 and g_2' see all of the polygon. If g_2 is to the left of $\mathcal{C}$, g_2' sees less of the pockets than g_2, so g_1 and g_2 can together see everything. If g_2 is to the right of $\mathcal{C}$, g_2 sees less of the pockets than g_2' and neither the top nor the bottom pocket are completely guarded by g_1 and g_2. For any higher placement of g_1 even less of the top pocket is guarded and for any lower placement of g_1 even less of the bottom pocket is guarded. Thus, there exists no placement of g_1 such that both pockets are completely guarded by g_1 and g_2. We summarize our reasoning in the following observation.

Observation 1 *Consider a fixed position of g_2 on or to the right of the segment bd. There exists a position of g_1 on l such that the entire polygon is seen by g_1 and g_2 if and only if g_2 lies on or to the left of the curve $\mathcal{C}$.*

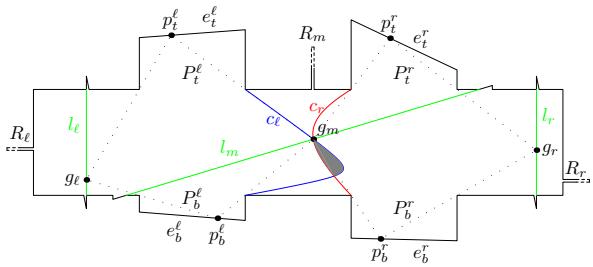


Figure 4: The polygon $\mathcal{P}$.

Restricting a Guard to a Single (Irrational) Point. For this paragraph, let us consider the polygon sketched in Figure 4, together with additional labels and information. The three guards g_ℓ, g_m, g_r are forced by the triangular pockets to lie on the three green line segments l_ℓ, l_m, l_r, respectively. Additionally, the three rectangular pockets R_ℓ, R_m, R_r force the guards to lie within short intervals within each line segment. With these restrictions, we can show that for the three guards to see the whole polygon, it must hold that the guards g_ℓ and g_m can together see the left quadrilateral pockets P_t^ℓ and P_b^ℓ, and the guards g_m and g_r can together see the right quadrilateral pockets P_t^r and P_b^r.

Then, the curve c_ℓ bounds from the right the feasible region for the guard g_m, such that g_ℓ and g_m can together see the left pockets P_t^ℓ and P_b^ℓ. Similarly, the curve c_r bounds from the left the feasible

region for the guard g_m, such that g_r and g_m can together see the right pockets P_t^r and P_b^r. Thus, the only way that g_ℓ, g_m, and g_r can see the whole polygon is when g_m is within the grey region, between c_ℓ and c_r. Our idea is to define the line l_m so that it contains an intersection point of c_ℓ and c_r, and it does not enter the interior of the grey region. A simple computation with some computer algebra system outputs equations defining the two curves c_ℓ, c_r. It can be checked, even by hand, that the point p lies on both curves, and also on the line l_m. Further p turns out to be irrational, see the longer versions for the exact values and computations [1]. Therefore, p is a feasible (and at the same time irrational) position for the guard g_m. Moreover, by plotting c_ℓ, c_r, and l_m in $\mathcal{P}$ as in Figure 4, we get an indication that as we traverse l_m from left to right, at the point p we exit the area where g_m and g_l can guard together the two left pockets, and at the same time we enter the area where g_m and g_r can guard together the two right pockets. Thus, the only feasible position for the guard g_m is the irrational point p.

References

[1] Mikkel Abrahamsen, Anna Adamaszek, and Tillmann Miltzow. Irrational Guards are Sometimes Needed. In *SoCG to appear*, 2017. available at `http://arxiv.org/abs/1701.05475`.

[2] Saugata Basu, Richard Pollack, and Marie-Françoise Roy. *Algorithms in real algebraic geometry*. Springer-Verlag Berlin Heidelberg.

[3] Patrice Belleville. Computing two-covers of simple polygons. Master's thesis, McGill University, 1991.

[4] Édouard Bonnet and Tillmann Miltzow. Parameterized hardness of art gallery problems. In *24th Annual European Symposium on Algorithms (ESA)*, pages 19:1–19:17, 2016. Full version available at `https://arxiv.org/abs/1603.08116`.

[5] Björn Brodén, Mikael Hammar, and Bengt J. Nilsson. Guarding lines and 2-link polygons is APX-hard. In *Proceedings of the 13th Canadian Conference on Computational Geometry (CCCG)*, pages 45–48, 2001.

[6] Vasek Chvatal. A combinatorial theorem in plane geometry. *Journal of Combinatorial Theory, Series B*, 18(1):39–41, 1975.

[7] Pedro Jussieu de Rezende, Cid C. de Souza, Stephan Friedrichs, Michael Hemmer, Alexander Kröller, and Davi C. Tozoni. Engineering art galleries. In *Algorithm Engineering: Selected Results and Surveys*, LNCS, pages 379–417. Springer, 2016.

[8] Alon Efrat and Sariel Har-Peled. Guarding galleries and terrains. *Inf. Process. Lett.*, 100(6):238–245, 2006.

[9] Stephan Eidenbenz, Christoph Stamm, and Peter Widmayer. Inapproximability results for guarding polygons and terrains. *Algorithmica*, 31(1):79–113, 2001.

[10] Sándor Fekete. Dagstuhl Seminar 11111, Computational Geometry, March 13 – 18 , 2011. *Organized by Pankaj Kumar Agarwal and Kurt Mehlhorn and Monique Teillaud*.

[11] Steve Fisk. A short proof of Chvátal's watchman theorem. *J. Comb. Theory, Ser. B*, 24(3):374, 1978.

[12] Stephan Friedrichs, Michael Hemmer, James King, and Christiane Schmidt. The continuous 1.5D terrain guarding problem: Discretization, optimal solutions, and PTAS. *Journal of Computational Geometry*, 7(1):256–284, 2016.

[13] Erik Krohn and Bengt J. Nilsson. Approximate guarding of monotone and rectilinear polygons. *Algorithmica*, 66(3):564–594, 2013.

[14] D. T. Lee and Arthur K. Lin. Computational complexity of art gallery problems. *IEEE Transactions on Information Theory*, 32(2):276–282, 1986.

[15] Joseph O'Rourke. *Art Gallery Theorems and Algorithms*. Oxford University Press, 1987.

[16] Joseph O'Rourke and Kenneth Supowit. Some NP-hard polygon decomposition problems. *IEEE Transactions on Information Theory*, 29(2):181–190, 1983.

[17] Günter Rote. EuroCG open problem session, 2011. See the personal webpage of Günter Rote: `http://page.mi.fu-berlin.de/rote/Papers/slides/Open-Problem_artgallery-Morschach-EuroCG-2011.pdf`.

[18] Dietmar Schuchardt and Hans-Dietrich Hecker. Two NP-hard art-gallery problems for ortho-polygons. *Math. Log. Q.*, 41:261–267, 1995.

[19] Thomas C. Shermer. Recent results in art galleries. *Proceedings of the IEEE*, 80(9):1384–1399, 1992.

[20] Ana Paula Tomás. Guarding thin orthogonal polygons is hard. In *Fundamentals of Computation Theory*, pages 305–316. Springer, 2013.

[21] Jorge Urrutia. Art gallery and illumination problems. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, chapter 22, pages 973–1027. Elsevier, 2000.

Illuminating polygons by edge-aligned floodlights of uniform angle (Brocard illumination)*

Carlos Alegría-Galicia † David Orden ‡ Carlos Seara § Jorge Urrutia ¶

Abstract

An α-*floodlight* is a light source that illuminates a wedge of the plane bounded by two rays ℓ, r emanating from a point x, in such way that r is obtained by rotating ℓ around x in the clockwise direction by an angle of α. The rays ℓ, r and the point x are known respectively, as the *left ray*, the *right ray*, and the *apex* of the α-floodlight. Given a simple polygon P with vertices $v_0, \ldots, v_{n-1}$ labelled in the clockwise order around the boundary of P, we say that an α-floodlight f_i is an *edge-aligned* α-*floodlight* on P, if its apex lies on v_i and its left ray contains the edge $\overline{v_i v_{i+1}}$. See Figure 1. Let $\mathcal{F}(\alpha)$ be the set $\{f_0, \ldots, f_{n-1}\}$ of edge-aligned α-floodlights on P. The *Brocard Illumination* problem for P consists on finding the smallest angle α such that $\mathcal{F}(\alpha)$ illuminates the interior of P. We present algorithms to solve the Brocard Illumination problem when P is convex or an arbitrary simple polygon using respectively, $\Theta(n)$ time and space, and $O(n^3 \log^2 n)$ time and $O(n^3)$ space. If P is a triangle, finding α is equivalent to finding the Brocard point of P.

1 Introduction

Let P be a simple polygon and $\mathcal{F}(\alpha)$ be the set of edge-aligned α-floodlights on P as defined above. We denote by ℓ_i and r_i respectively, the left ray and the right ray of f_i. Note that as α is increased, ℓ_i remains fixed (as it contains $\overline{v_i v_{i+1}}$) while r_i rotates in the clockwise direction anchored at v_i. Therefore, all the rays $r_0, \ldots, r_{n-1}$ rotate at the same speed, and the intersection point of any two rays r_i and r_j describes a circular arc contained on a circle $c_{(i,j)}$ passing through v_i and v_j that we call the *adjoint circle* of v_i and v_j. It is not hard to see that $c_{(i,i+1)}$ is tangent to $\overline{v_i v_{i+1}}$ at v_{i+1} and, for $|i-j| \neq 1$, $c_{(i,j)}$ passes through the intersection point of the lines containing $\overline{v_i v_{i+1}}$ and $\overline{v_j v_{j+1}}$. For simplicity, we denote $c_i = c_{(i,i+1)}$. See Figure 1 (left).

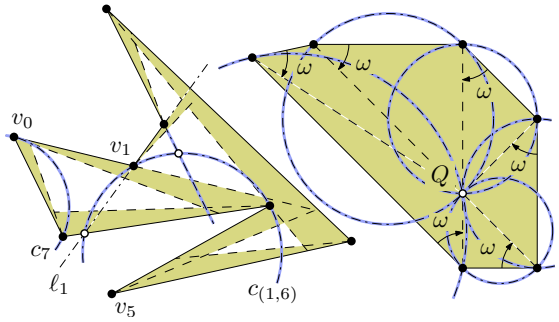


Figure 1: The Brocard Illumination of a simple polygon (left) and a Brocard hexagon (right). Adjoint circles are highlighted with blue.

A *Brocard polygon* [2] is a convex polygon having a unique point Q in its interior such that $\angle Q v_i v_{i+1} = \angle Q v_{i+1} v_{i+2} = \alpha_b$ for all $0 \leq i < n$, where subindices are taken modulo n. The point Q and the angle α_b are known respectively, as the *Brocard point* and the *Brocard angle* of the polygon. For P being a Brocard polygon, all the adjoint circles $c_0, \ldots, c_{n-1}$ intersect in Q. That is, Q is the common intersection point of all the rotating rays of the floodlights in $\mathcal{F}(\alpha_b)$ and, thus, it is the last point in the interior of P being illuminated, so that it gives the solution of the Brocard Illumination problem. See Figure 1 (right).

It is known that all harmonic polygons are Brocard polygons [4], as well as all regular polygons and all triangles. It requires $O(n)$ time and space to verify if P belongs to one of these classes of polygons and, in such case, we can clearly solve the Brocard Illumination problem in constant time. In this paper we present algorithms to solve the Brocard Illumination problem for two more general classes of polygons: When P is convex or an arbitrary simple polygon. For each class our algorithms require respectively, $\Theta(n)$ time and space, and $O(n^3 \log^2 n)$ time and $O(n^3)$ space. As

*Research of all the authors is partially supported by project H2020-MSCA-RISE project 73499 - CONNECT.

†Posgrado en Ciencia e Ingeniería de la Computación, Universidad Nacional Autónoma de México, calegria@uxmcc2.iimas.unam.mx.

‡Departamento de Física y Matemáticas, Universidad de Alcalá, Spain, david.orden@uah.es. Research supported by MINECO Projects MTM2014-54207 and TIN2014-61627-EXP and by TIGRE5-CM Comunidad de Madrid Project S2013/ICE-2919.

§Departament de Matemàtiques, Universitat Politècnica de Catalunya, Spain, carlos.seara@upc.edu. Research supported by the projects Gen. Cat. DGR 2014SGR46 and MINECO MTM2015-63791-R.

¶Instituto de Matemáticas, Universidad Nacional Autónoma de México, urrutia@matem.unam.mx. Research supported by SEP-CONACYT 80268 and PAPPIIT IN102117 Programa de Apoyo a la Investigación e Innovación Tecnológica UNAM.

far as we know, there are no previous studies about the Brocard Illumination of a polygon, the most related ones being [5–9]. These consider sets of vertex floodlights of uniform angle, but they do not require the floodlights to be edge-aligned. See also [2, 10].

2 Convex polygons

In this section we assume that P is a convex polygon. Let $P(\alpha) = P \setminus (f_0 \cup \cdots \cup f_{n-1})$ be the convex polygon in the interior of P that is not yet illuminated by the floodlights in $\mathcal{F}(\alpha)$ for a given α, and $x_{(i,j)}$ denote the intersection point of r_i and r_j. The circles $c_{(i,j)}$ and $c_{(j,k)}$ intersect at two points, one of which is v_j. We denote with $a_{(i,j,k)}$ the second intersection point of $c_{(i,j)}$ and $c_{(j,k)}$, and with $\alpha_{(i,j,k)}$ the angle $\angle a_{(i,j,k)} v_i v_{i+1} = \angle a_{(i,j,k)} v_j v_{j+1}$. For simplicity, we denote $x_i = x_{(i,i+1)}$, $a_i = a_{(i,i+1,i+2)}$, and $\alpha_i = \alpha_{(i,i+1,i+2)}$.

Note that $P(0) = P$ and $P(\alpha)$ shrinks as α grows up to the angle α_b of Brocard Illumination of P. It is known that $P(\alpha_b)$ is a degenerate polygon formed by a single point or a line segment [1]. In the former case, the point is the common intersection of the rotating rays of at least three floodlights in $\mathcal{F}(\alpha_b)$ with non-parallel left rays. In the latter case, the line segment connects two vertices v_i and v_j, since at α_b the right rays of f_i and f_j meet at the line segment $\overline{v_i v_j}$.

Consider the set $\mathcal{F}(\alpha)$ and the polygon $P(\alpha)$ as we increment α starting at $\alpha = 0$. Assume for simplicity that no pair of edges of P are parallel to each other, and that α_0 is the smallest angle in $\alpha_0, \ldots, \alpha_{n-1}$. When $0 \le \alpha < \alpha_0$, the vertices of $P(\alpha)$ are the points $x_0, \ldots, x_{n-1}$ and, for $0 \le i < n$, the points x_i and x_{i+1} move towards a_i along c_i and c_{i+1}, respectively. See Figure 2.

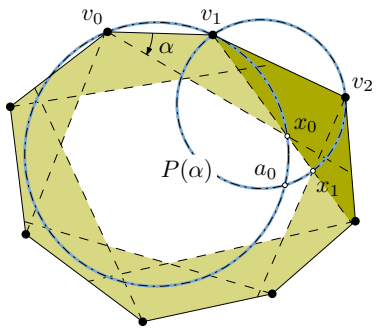


Figure 2: As α grows, x_0 and x_1 converge to a_0.

At $\alpha = \alpha_0$ we have that $x_0 = x_1 = a_0$, that is, the edge $\overline{x_0 x_1}$ degenerates into a point. Note that r_1 does not participate on the boundary of $P(\alpha_0)$ and, from here on, f_1 can be "turned off" without affecting the area of P illuminated by the remaining floodlights. In these conditions we say that α_0 is a *blackout event* and f_1 is discarded. See Figure 3.

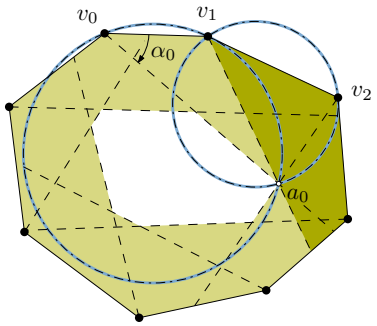


Figure 3: A blackout event at α_0.

Let ε be an angle such that the blackout event after α_0 arises at an angle greater than $\alpha_0 + \varepsilon$. The vertices of $P(\alpha_0 + \varepsilon)$ are $x_{(0,2)}, x_2, \ldots, x_{n-1}$, the point $x_{(0,2)}$ moves towards $a_{(0,2,3)}$ along $c_{(0,2)}$, and the next blackout event is the smallest angle in the set $\alpha_{(0,2,3)}, \alpha_2, \ldots, \alpha_{n-2}, \alpha_{(n-1,0,2)}$. See Figure 4.

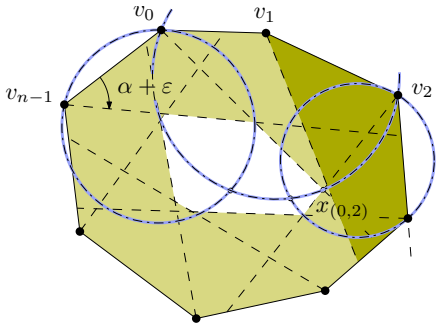


Figure 4: The floodlights in $\mathcal{F}(\alpha + \varepsilon)$.

Thus, for every blackout event two elements are removed and one new element is added to the set of adjoint circles. Therefore, at each event the number of candidate blackout events, as well as the vertices of $P(\alpha)$, are reduced by one. If we keep on repeating this process, we will eventually have a set of candidate blackout events all equal to the angle of Brocard Illumination α_b, where $P(\alpha_b)$ is a single point given by the common intersection of the remaining adjoint circles.

Lemma 1 *Let n_α be the number of vertices of $P(\alpha)$. We can find out whether the floodlights in $\mathcal{F}(\alpha)$ illuminate the interior of P in $O(n_\alpha)$ time and space.*

We now describe the algorithm to solve the Brocard Illumination problem for a convex polygon.

1. **Initialization.** Initialize a circular linked list $\mathcal{L}$ to contain pairs of the form (x, θ), where x is a point in $\mathbb{R}^2$ and θ is an angle in the unit circle. Traverse the vertices of P in clockwise circular order and, for each vertex v_i, add the entry (v_i, α_i) to $\mathcal{L}$. At the end of this process $\mathcal{L}$ will contain the pairs

$$(v_0, \alpha_0) \to \cdots \to (v_{n-1}, \alpha_{n-1}) \to (v_0, \alpha_0).$$

Note that the set of angles of the elements in $\mathcal{L}$ form the unsorted set of candidate blackout events induced by the set of vertices of P.

2. **Recursive binary search.** Repeat the following steps until the angles of all the elements in $\mathcal{L}$ have the same value, or $\mathcal{L}$ contains at most three elements:

(a) Compute the median α_m of the set of angles of the elements in $\mathcal{L}$. If there are repeated angles, choose α_m so that all of them lie in the same part of the bipartition induced by α_m.

(b) For a given α, let

$$\mathcal{F}^-(\alpha) = \{f_i \in \mathcal{F}(\alpha) \mid (v_i, \theta) \in \mathcal{L} \text{ and } \theta \leq \alpha\},$$
$$\mathcal{F}^+(\alpha) = \{f_i \in \mathcal{F}(\alpha) \mid (v_i, \theta) \in \mathcal{L} \text{ and } \theta > \alpha\}.$$

Note that the region of P illuminated by the floodlights in $\mathcal{F}^-(\alpha)$ is contained in the region illuminated by the floodlights in $\mathcal{F}^+(\alpha)$. We can therefore discard one of the partitions of $\mathcal{L}$ induced by α_m (as it does not participate in the solution) as follows.

i. If the floodlights in $\mathcal{F}^-(\alpha_m)$ illuminate the interior of P, then α_m is greater than the angle of Brocard illumination. We therefore remove from $\mathcal{L}$ the elements whose points are apexes of floodlights in $\mathcal{F}^+(\alpha_m)$.

ii. If the floodlights in $\mathcal{F}^-(\alpha_m)$ do not illuminate the interior of P, then α_m is smaller than the angle of Brocard illumination. We therefore remove from $\mathcal{L}$ the elements whose points are apexes of floodlights in $\mathcal{F}^-(\alpha_m)$.

As $\mathcal{L}$ is a linked list, while traversing $\mathcal{L}$ each element can be removed in constant time. Moreover, by Lemma 1 we require linear time to check if the floodlights in any subset of $\mathcal{F}(\alpha)$ illuminate P for a given α. Therefore, a linear run suffices to remove from $\mathcal{L}$ the elements required at either condition i or ii.

(c) Traverse the remaining elements of $\mathcal{L}$ and, for each triplet $((v_i, \alpha), (v_j, \beta), (v_k, \gamma))$ of consecutive elements, set $\alpha = \alpha_{(i,j,k)}$.

3. **Process parallel edges.** Compute the angle $\angle v_j v_i v_{i+1} = \angle v_i v_j v_{j+1}$ for every pair (f_i, f_j) of floodlights whose left rays are parallel to each other. Keep the smallest of such angles in a variable α_p. This can be done in $O(n)$ time and space using rotating calipers.

4. **Report solution.** Report as the Brocard Illumination angle of P the smallest of α_p and the angle of the remaining elements in $\mathcal{L}$.

Steps 1, 3 and 4 spend linear time and space. Step 2 also requires $O(n)$ time and space, as it is a binary search over an unsorted set and its stop condition, as well as Steps 2a to 2c, spend $O(n)$ time each. We therefore obtain the following result.

Theorem 2 *The Brocard illumination problem for a convex polygon can be solved in $\Theta(n)$ time and $O(n)$ space.*

3 Simple polygons

In this section P is a simple (not necessarily convex) polygon. We start by generalizing the observations made in Section 2 in the following Lemma.

Lemma 3 *The Brocard illumination angle for P is determined by three rays r_i, r_j, r_k meeting at a point (see Figure 1, left), or by two rays r_i and r_j meeting at the line segment $\overline{v_i v_j}$ (see Figure 5).*

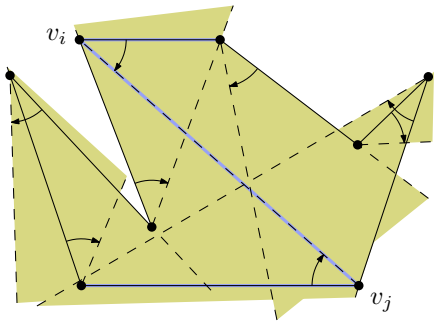


Figure 5: A Brocard illumination angle given by two edge-aligned floodlights with parallel left rays.

Based on Lemma 3 we also generalize the binary-search approach used in Section 2. We sketch next the resulting algorithm.

1. **Candidate Angles.** Based on the conditions from Lemma 3, compute the set $\mathcal{A}$ of $O(n^3)$ candidate angles of Brocard illumination.

(a) For every pair (v_i, v_j) such that v_j is visible from v_i, add to $\mathcal{A}$ the angle $\angle v_j v_i v_{i+1}$ if $\overline{v_i v_{i+1}}$ is parallel to $\overline{v_j v_{j+1}}$ or v_j is reflex. These candidate angles cover the cases where two rays r_i, r_j meet at $\overline{v_i v_j}$, or three rays r_i, r_j, r_k meet at a point and at least one of them passes through a second vertex of P (see Figure 1, left, where the last illuminated point in the interior of P is found by intersecting r_3 and $c_{(1,6)}$).

(b) Add to $\mathcal{A}$ the angles given by the intersection point $Q_{(i,j,k)}$ of all triplets of circles $c_{(i,j)}$, $c_{(j,k)}$, and $c_{(j,k)}$ such that $Q_{(i,j,k)}$ lies in the interior of P. These candidate angles cover the case where three right rays meet at a point and none of them passes through a second vertex of P.

2. **Process visibility polygons.** Compute the visibility polygon P_i of every vertex v_i, and the set $\mathcal{S}$ of intersection points of all the line segments inside P induced by the polygons $P_0, \ldots, P_{n-1}$. It is not hard to see that the number of such intersection points for

every pair (P_i, P_j) is in $O(n)$ and therefore, there are $O(n^3)$ elements in $\mathcal{S}$. To finalize this step, add the vertices of P to $\mathcal{S}$ and then sort the elements of $\mathcal{S}$ in increasing order along the x-axis.

3. **Binary search.** Sort the elements of $\mathcal{A}$ in increasing order and perform a binary search on the elements of $\mathcal{A}$. At each iteration, find out whether the floodlights in $\mathcal{F}(\alpha)$ illuminate the interior of P for the current value of α, to decide in which of the two parts of $\mathcal{A}$ to perform the recursive search.

We define the *α-visibility polygon* of v_i as the simple polygon $P_i(\alpha)$ formed by the points inside P that are visible from v_i through f_i. To find out whether $\mathcal{F}(\alpha)$ illuminates the interior of P, compute the union of the polygons $P_0(\alpha), \ldots, P_{n-1}(\alpha)$ for the given value of α by performing a plane-sweep.

(a) Use the elements of the (sorted) set $\mathcal{S}$ as stop events.

(b) Maintain the intersections of the sweeping line, P, and the set $P_0(\alpha), \ldots, P_{n-1}(\alpha)$. These intersections form a set of intervals on the sweeping line, representing the interior of P and the illuminated area of the floodlights in $\mathcal{F}(\alpha)$ for the current value of α. Note that the intervals are not disjoint to each other and, in fact, several intervals share an endpoint.

(c) Discard the events that correspond to points not belonging to an α-visibility polygon for the current value of α, as they do not belong to the union of the polygons $P_0(\alpha), \ldots, P_{n-1}(\alpha)$.

(d) Based on the intervals over the sweep line, decide whether the current value of α suffices to illuminate the interior of P.

Computing $\mathcal{A}$ requires $O(n^3 \log n)$ time and $O(n^3)$ space, as Step 1a can be computed using the vertex visibility graph of P, and in Step 1b we can use ray shooting to find out if the segments $\overline{Q_{(i,j,k)}v_x}$, $x \in \{i, j, k\}$, lie in the interior of P, spending $O(\log n)$ time per triplet. Step 2 clearly requires $O(n^3 \log n)$ time and $O(n^3)$ space. The binary search of Step 3 is the most expensive part of the algorithm: At each of the $O(\log n)$ iterations, we perform a plane-sweep on a sorted set of $O(n^3)$ stop events, and we take $O(\log n)$ time processing each event. That is, we require $O(n^3 \cdot \log n \cdot \log n) = O(n^3 \log^2 n)$ time and $O(n^3)$ space.

Theorem 4 *The Brocard Illumination angle of a simple polygon can be computed in $O(n^3 \log^2 n)$ time and $O(n^3)$ space.*

4 Final remarks

For a convex polygon, a bound of $\frac{\pi}{2}$ on the angle of Brocard illumination is easy to obtain. Three floodlights are always sufficient and sometimes necessary to illuminate the polygon using at most $\frac{3}{2}\pi$ of total power. In a Brocard polygon, it is known that $\alpha_b \leq \frac{\pi}{2} - \frac{\pi}{n}$ [3] (the equality is reached when the polygon is regular), for a total power of at most $\pi(\frac{3}{2} - \frac{3}{n})$.

It is known that a simple polygon can always be illuminated using vertex π-floodlights. However, if the floodlights are edge-aligned it is not hard to prove that an aperture angle of π does not always suffice to illuminate the interior of the polygon. As a simple polygon requiring a linear number of edge-aligned floodlights is not hard to obtain, in this case we require a total power greater than πn.

Our approach to compute the Brocard Illumination problem for simple polygons is based on the efficient computation of the union of the set of α-visibility polygons for a given value of α. As we do not require to compute the union of those polygons (as opposed to decide if such union illuminates the interior of P), we believe that our $O(n^3 \log^2 n)$ time bound can be improved, although it does not seem to be an easy problem to solve.

References

[1] A. Ben-Israel and S. Foldes. Complementary half-spaces and trigonometric Ceva-Brocard inequalities for polygons. *Mathematical Inequalities & Applications.* Vol. 2(2), (1999), 307–316.

[2] A. Bernhart. Polygons of pursuit. *Scripta Mathematica.* Vol. 24(1), (1959), 23–50.

[3] Á. Besenyei. The Brocard Angle and a geometrical gem from Dmitriev and Dynkin. *The American Mathematical Monthly.* Vol. 122(5), (2015), 495–499.

[4] J. Casey. A sequel to the first six books of the elements of Euclid. *Dublin University Press.* (1888).

[5] F. Contreras, J. Czyzowicz, N. Fraiji, and J. Urrutia. Illuminating triangles and quadrilaterals with vertex floodlights. *10th Canadian Conference on Computational Geometry.* (1998).

[6] V. Estivill-Castro, J. O'Rourke, J. Urrutia, and D. Xu. Illumination of polygons with vertex lights. *Information Processing Letters.* Vol. 56, (1995), 9–13.

[7] D. Ismailescu. Illuminating a convex polygon with vertex lights. *Periodica Mathematica Hungarica.* Vol. 57(2), (2008), 177–184.

[8] J. O'Rourke, T. Shermer, and I. Streinu. Illuminating convex polygons with vertex floodlights. *7th Canadian Conference on Computational Geometry.* (1995).

[9] C. Tóth. Art galleries with guards of uniform range of vision. *Computational Geometry: Theory and Applications.* Vol. 21, (2002), 185–192.

[10] J. Urrutia. Art gallery and illumination problems. *Handbook of Computational Geometry.* (2000), 973–1027.