Feb
18
2021

YAZD UNIVERSITY

4TH

Iranian Conference on Computational Geometry

ICCG 2021

Proceedings

دانشگاه یزد

ISC
Islamic World Science Citation Center
99201-97366

4th Iranian Conference on

Computational Geometry

(ICCG 2021)

Proceedings

Yazd University
Yazd, Iran, February 18, 2021

Foreword

The fourth Iranian Conference on Computational Geometry was held online on February 18, 2021 hosted by the department of Mathematical Sciences of Yazd University, in the world heritage city in UNESCO, Yazd. The goal of this annual, international conference is to bring together students and researchers from academia and industry, in order to promote research in the fields of combinatorial and computational geometry. Because of the COVID-19 situation, the whole conference held online.

This volume of proceedings contains a selection of eleventh refereed papers that were presented during the conference, in three sections. I would like to thank PC chairs Hamid Zarrabi-Zadeh and Ahmad Biniaz, all the PC members, and members of the local organizing committee. I also want to thank the sponsors: Yazd University for financial supports, the RayaGraph company for supporting the conference website, and Islamic World Science Citation Center (ISC) for indexing the conference (#ISC 99201-97366).

Mohammad Farshi

Invited Speaker

Anil Maheshwari Carleton University, Canada

General Chair

Mohammad Farshi Yazd University

Program Committee

Mohammad Ali Abam	Sharif University of Technology
Hugo Akitaya	Carleton University
Elena Arseneva	St. Petersburg State University
Fatemeh Baharifard	IPM
Therese Biedl	University of Waterloo
Ahmad Biniaz	University of Windsor (co-chair)
Mark de Berg	Eindhoven University of Technology
Stephane Durocher	University of Manitoba
Amin Gheibi	Amirkabir University of Technology
Mahdieh Hasheminezhad	Yazd University
Shahin Kamali	University of Manitoba
Wolfgang Mulzer	Freie Universitat Berlin
Zahed Rahmati	Amirkabir University of Technology
Don Sheehy	North Carolina State University
Michiel Smid	Carleton University
Andre van Renssen	The University of Sydney
Alireza Zarei	Sharif University of Technology
Hamid Zarrabi-Zadeh	Sharif University of Technology (co-chair)

Local Organizers

Mahdieh Hasheminezhad	Yazd University
Mohammad Farshi	Yazd University
Mohammad Javad Hekmat Nasab (Website)	Amirkabir University of Technology

Conference Program

Thursday February 18

Session 1

Session 2

Index of Authors

Guarding Polyominoes under k-Hop Visibility or Minimum k-Dominating Sets in Grid Graphs*

Christiane Schmidt†

Abstract

We consider a guarding problem in polyominoes: a unit-square guard sees all unit squares within distance k in the dual grid graph. The problem is equivalent to the minimum k-dominating set problem in grid graphs. We prove NP-completeness of this problem in polyominoes with holes for $k \in \{1, 2\}$ and provide lower bounds for all k and matching upper bounds for $k \in \{1, 2\}$ of $\lfloor \frac{m}{k+1} \rfloor$ on the number of guards in any polyomino.

1 Introduction

In the classical art gallery problem (AGP), we aim to place guards in a polygon, such that every point of the polygon is visible to at least one guard. Visibility is defined by analogy to human vision: two points $u, v \in P$ see each other if the line segment $\overline{u, v}$ is fully contained in P. Various variants for the classical AGP have been considered (varying both the capabilities of the guards and the environment to be guarded), and usually we are interested in two types of questions:

1. Can we compute the minimum cardinality guard set for a polygon P?
2. What are lower/upper bounds on the number of guards needed to cover a polygon from a given class?

For the classical AGP, question (1) was answered with several complexity results: NP-hardness was proven for different problem variants ([1, 2]). Answers to question (2) are often referred to as "Art Gallery theorems". Chvátal [3] provided the first such result: a tight bound of $\lfloor \frac{n}{3} \rfloor$ for simple polygons.

Here, we consider a guarding problem motivated from serving a city with carsharing (CS) stations: the demand is given in a granularity of (square) cells, and we assume that customers are willing to walk a certain distance to a CS stations—a simplified assumption, which we can substitute by obtaining demand for given stations using a multi-agent transport simulation, MATSim[1]. We also assume that this walking-range bound is the same for the complete city. Then, we aim to place as few CS stations as possible to serve the complete city for a given maximum walking range. We represent the city as a polyomino, potentially with holes and only walking within the boundary is possible—e.g., in Stockholm holes usually represent water bodies, which pedestrians cannot cross. This yields a special type of "visibility" for a station: all unit squares of the polyomino reachable when walking inside the polyomino for at most the given walking range.

Guarding polyominoes has been considered by Biedl et al. [4], who considered different models of visibility. ($u \in P$ sees $v \in P$ if the axis-parallel rectangle spanned by u, v is fully contained in P). They provided both NP-hardness results and art gallery theorems in terms of the number of unit squares of the polyomino, m. NP-hardness for another type of visibility (rectangle visibility) was provided in [5].

An equivalent formulation of our problem is in terms of the minimum k-dominating set problem (MkDSP): find a minimum cardinality $D_k \subseteq V(G)$, such that each graph vertex is connected to a vertex in D_k with a path of length at most k. We aim to solve MkDSP in grid graphs (the dual graph of a polyomino). The minimum dominating set problem is NP-complete [6], hence, the MkDSP is clearly NP-complete in general graphs.

Notation. A *polyomino* is a connected polygon P in the plane formed by joining together $|P| = m$ unit squares on the square lattice. The dual graph G_P of a polyomino has a vertex for each unit square and $\{u, v\} \in E(G_P)$ if unit squares u, v are adjacent; G_P is a *grid graph*. P is *simple* if it has no holes, that is, if every minimal cycle in the dual grid graph is a 4-cycle.

A unit square $v \in P$ is *k-hop-visible* to a unit square $u \in P$ if the shortest path from u to v in the dual grid graph of P, G_P, has length at most k. See Figure 1 for an example. Note that for $k \geq 2$ this in particular includes the ability to look around a corner of the polyomino. A *witness* placed at unit square u vouches that at least one *guard* has to be placed in its k-hop-visibility region.

Minimum k-hop Guarding Problem (MkGP).
Given: A polyomino P, a range k.
Find: The minimum number cardinality unit-square guard cover in P under k-hop visibility.

*This work is supported by grant 2018-04101 (`https://tinyurl.com/EkoCS-Trans`) from Sweden's innovation agency VINNOVA.

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

[1] `matsim.org`

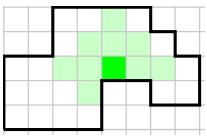


Figure 1: A polyomino P (black), a unit-square guard g (green) and its visibility region (light green), $k = 2$.

2 NP-Completeness

We show NP-completeness of the MkGP in polyominoes with holes and $k = 2$. We reduce from PLANAR 3-SAT.

An instance F of the PLANAR 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 iff it contains at least one true literal, and the formula F is true iff 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, see Knuth and Raghunathan [7].

We turn the rectilinear embedding of G into a polyomino: we represent the variables, clauses and edges by pieces of a polyomino that needs to be guarded. We construct **variable gadgets** as shown in turquoise in Figure 2(a). There exist exactly two feasible placements of guards for the variable loop gadget, shown in blue and red in Figure 2(b) and (c) and corresponding to a truth setting of "true" and "false", respectively.

The initial truth value is propagated by a **wire gadget**. In Figure 2 we show a wire gadget for the case that the variable appears in a clause in dark blue, and for the case that the negated variable appears in the clause in dark red. We note that a wire can easily be bend by 90°. We extend the width of the variable gadget to connect to further wire gadgets.

The **clause gadget** is shown in Figure 3(a): Wires connecting to the three variables connect to it from the top, the right, and the bottom. The clause gadget can be covered with exactly two additional guards if one (see Figure 3(d)-(f)), two (see Figure 3(g)-(i)) or all (see Figure 3(j)/(k)) variables have a truth setting fulfilling the clause. If all variables have a truth setting not fulfilling the clause (see Figure 3(b)), three additional guards are needed to cover the clause gadget: The k-hop visibility regions of the three colored witnesses in Figure 3(b) are pairwise disjoint, hence, at least three guards are necessary—and sufficient, see Figure 3(c).

Thus, we solve the MkGP optimally iff 1-3 variables per clause have a truth setting fulfilling the clause, that is, iff the original PLANAR 3-SAT formula F is satisfiable. The reduction is possible in polynomial time. Moreover, given a set of unit-square guards, we can easily determine the k-hop visibility region of all guards and check whether all unit squares are covered. Hence, the MkGP is in NP. This yields:

Theorem 1 *MkGP is NP-complete for $k = 2$ in polyominoes with holes.*

A similar variable and corridor gadget construction and the clause from Fig. 4 yield NP-completeness also for $k = 1$ (due to space restrictions without proof):

Theorem 2 *MkGP is NP-complete for $k = 1$ in polyominoes with holes.*

3 Art Gallery Theorems

In this section, we provide lower bounds for all k (Theorem 3) and matching upper bounds for $k \in \{1, 2\}$ (Theorem 4) on the number of guards necessary to cover polyominoes under k-hop visibility.

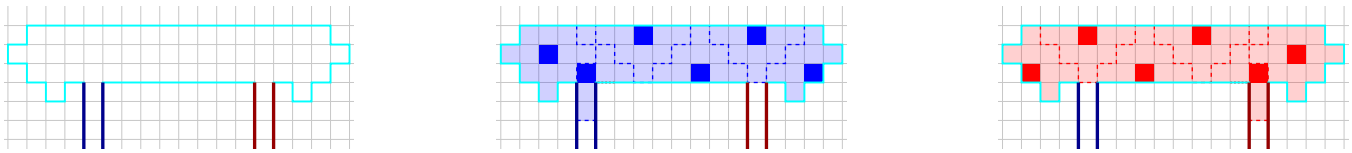
Theorem 3 *There exist simple polyominoes with m unit squares that require $\lfloor \frac{m}{k+1} \rfloor$ guards to cover their interior under k-hop visibility.*

Proof. We construct a double-comb like polyomino: we alternately add teeth of length k to the top and bottom of a row of unit squares (the shaft), see Figure 6 for the construction for $k = 1$ and $k = 2$. If m is not divisible by $k + 1$ we add $x = (m \mod k + 1)$ unit squares to the right of the shaft. Witnesses placed at the last unit square of each tooth (shown in pink in Figure 6) have disjoint k-hop visibility regions (of the shaft only the unit square to which the tooth is attached belongs to the k-hop visibility region), hence, we need one guard per witness. The x unit squares to the right of the shaft can be covered by the rightmost guard if placed in the shaft. Let t be the number of teeth, $m = t \cdot (k + 1) + x$, we need $t = \lfloor \frac{m}{k+1} \rfloor$ guards. □

Theorem 4 *$\lfloor \frac{m}{k+1} \rfloor$ guards are always sufficient and sometimes necessary to cover a polyomino with m unit squares under k-hop visibility for $k \in \{1, 2\}$.*

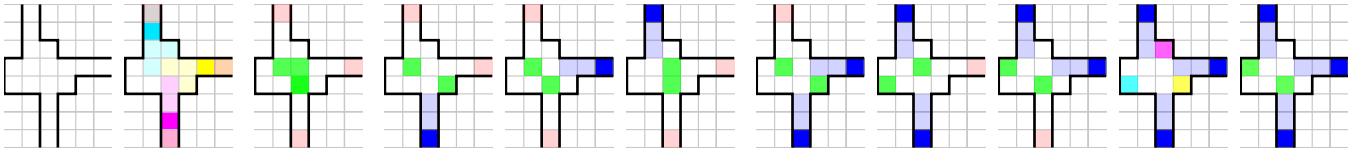
Proof. We need to show that $\lfloor \frac{m}{r+1} \rfloor$ guards are always sufficient. We give constructive proofs for $r \in \{1, 2\}$.

Case $k = 1$. Compute a maximum matching M in the (bipartite) dual grid graph of P, G_P. Every vertex in G_P that is not matched is adjacent to matched vertices only (otherwise we could extend M). For each matching edge $\{u, v\}$ unmatched vertices are adjacent to u or v only (otherwise, let u' and v' be an unmatched vertex adjacent to u and v, respectively, then $M \setminus \{u, v\} \cup \{u', u\} \cup \{v, v'\}$ is a larger matching than M). For each matching edge, we place a guard at the unit square of the vertex in G_P that is adjacent to unmatched vertices (if any, otherwise we choose one of the



(a) (b) (c)

Figure 2: (a) Variable gadget in turquoise, wire gadgets in dark blue (in case the variable appears in a clause) and dark red (in case the negated variable appears in a clause). We associate the solution in (b) and (c) with a truth setting of "true" and "false", respectively.



(a) (b) (c) (d) (e) (f) (g) (h) (i) (j) (k)

Figure 3: (a) Clause gadget. (b)-(k) The truth setting of the variables connected by the three variable corridors is shown in red/blue, where light-blue/light-red indicates the visibility region of a guard, a red/blue unit square indicates the guard's location. (b) All variables have a truth setting that does not fulfill the clause, then the three colored witnesses (visibility regions in lighter colors) cannot be covered by the same guard, hence, three guards are necessary, and sufficient (c). If one variable has a truth setting that does fulfill the clause, (d)-(f), and if two variables have a truth setting that does fulfill the clause, (g)-(i), two (green) guards suffice to cover the clause gadget. If all variables have a truth setting that does fulfill the clause, the three witnesses in (j) cannot all be covered with a single guard, the two green guards in (k) are sufficient to cover the clause gadget.

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

Figure 4: Clause gadget for $k = 1$. (a) If all variables have a truth setting not fulfilling the clause, the three pink witnesses cannot be covered by the same guards; three green guards are sufficient. (b) If one variable has a truth setting that fulfills the clause two green guards suffice. The other cases are omitted. Color usage as in Figure 3.

two vertices). This guard covers its matched neighbor and all unmatched neighbors. Hence, we placed at most $\lfloor \frac{m}{k+1} \rfloor = \lfloor \frac{m}{2} \rfloor$ guards to cover P.

Case $k = 2$. Again, we compute a maximum matching M in G_P. We build a graph G_M based on M: we create a vertex in G_M for each matching edge and each unmatched vertex in M ($V(G_M) = \{v_{\{u,u'\}} : \{u, u'\} \in M\} \cup \{v_u : u \in G_P \setminus M\}$). Two vertices v, v' in G_M are connected by an edge if:

- For $v = v_{\{u,u'\}}, v' = v'_{\{w,w'\}}$: if at least one of the edges $\{u, w\}, \{u, w'\}, \{u', w\}$ or $\{u', w'\}$ is in $E(G_P)$.
- For $v = v_{\{u,u'\}}, v' = v'_w$: if at least one of $\{u, w\}$ or $\{u, w'\}$ is in $E(G_P)$.

We compute a maximum matching M' in G_M. Each matching edge in M' represents three or four vertices of G_P. Each unmatched vertex in M' represents one or two vertices of G_P. Again, unmatched vertices are adjacent to at most one of the vertices per edge in M'. If an unmatched vertex is adjacent to more than one matched vertex, we assign it to one of them. In the remainder of this proof adjacent unmatched vertex/vertices refers to the *assigned* adjacent unmatched vertex/vertices only. We distinguish six cases, see Figure 5 for examples:

1. $e = \{v_{\{u,u'\}}, v_{\{w,w'\}}\} \in M'$ is not adjacent to any unmatched vertex in M': we have a path of length 4, and place a guard on one of the two vertices that are adjacent to two of the other three vertices. Hence, the single guard covers 4 unit squares.
2. $e = \{v_{\{u,u'\}}, v_{\{w,w'\}}\} \in M'$ is adjacent to unmatched vertices, all unmatched vertices adjacent to e represent one vertex from G_P. W.l.o.g. let the unmatched vertices be adjacent to $v_{\{u,u'\}}$:
   (a) $\{u, w\} \in G_P$ or $\{u, w'\} \in G_P$, but $\{u', w\} \notin G_P$ and $\{u', w'\} \notin G_P$: We place a guard on u, then u, u', w, w' and all unmatched vertices adjacent to $v_{\{u,u'\}}$ are covered. The single guard at u covers at least 5 unit squares.
   (b) $\{u, w\} \in G_P$ and $\{u', w'\} \in G_P$ (or $\{u', w\} \in G_P$ and $\{u, w'\} \in G_P$): A guard placed on u or u' covers u, u', w, w' and all unmatched vertices adjacent to $v_{\{u,u'\}}$. The single guard covers at least 5 unit squares.
3. $e = \{v_{\{u,u'\}}, v_{\{w,w'\}}\} \in M'$ is adjacent to unmatched vertices, some unmatched vertices adjacent to e represent two vertices from G_P. Let the unmatched vertices be adjacent to $v_{\{u,u'\}}$: We place two guards at u and u' and cover u, u', w, w' and all

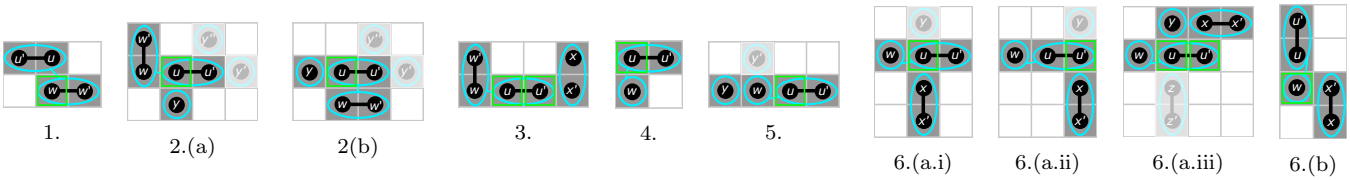


Figure 5: Cases from the proof of Lemma 4, $k = 2$. Vertices and M in G_P shown in black; vertices and M' in G_M shown in turquoise; guards shown in green. Optional unit squares are shown faded.

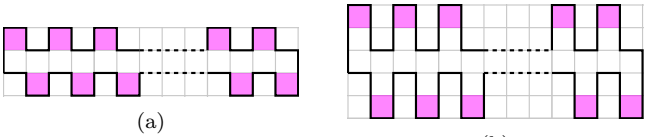


Figure 6: Lower bound construction for polyominoes that require $\lfloor \frac{m}{k+1} \rfloor$ guards under k-hop visibility for (a) $k = 1$, (b) $k = 2$. Witnesses are shown in pink.

unmatched vertices adjacent to $v_{\{u,u'\}}$. These two guards cover at least 6 unit squares: u, u', w, w' and at least one pair of vertices x, x', where $v_{\{x,x'\}}$ unmatched in M' and adjacent to $v_{\{u,u'\}}$. Hence, on average, each of the guards covers at least 3 unit squares.

4. $e = \{v_{\{u,u'\}}, v_w\} \in M'$ is not adjacent to any unmatched vertex in M', w.l.o.g. $\{u, w\} \in E(G_P)$: We place a guard on u, it covers 3 unit squares.
5. $e = \{v_{\{u,u'\}}, v_w\} \in M'$ is adjacent to unmatched vertices adjacent to e representing one vertex from G_P, w.l.o.g. $\{u, w\} \in E(G_P)$: We place a guard on u, which covers u, u' and w and all vertices adjacent to these three (independent of whether all are adjacent to $v_{\{u,u'\}}$ or to v_w). A single guard covers at least 4 unit squares.
6. $e = \{v_{\{u,u'\}}, v_w\} \in M'$ is adjacent to unmatched vertices, some of these represent two vertices from G_P, w.l.o.g. $\{u, w\} \in E(G_P)$:
   (a) The unmatched vertices are adjacent to $v_{\{u,u'\}}$:
      (i) For all unmatched vertices v_y, y is adjacent to u and for all unmatched vertices $v_{\{x,x'\}}$ x or x' is adjacent to u: We place a guard at u. This single guard covers at least 5 unit squares.
      (ii) For all unmatched vertices v_y, y is adjacent to u' and for all unmatched vertices $v_{\{x,x'\}}$ x or x' is adjacent to u': We place a guard at u'. This single guard covers at least 5 unit squares.
      (iii) We have at least one unmatched vertex for which one of the vertices in G_P it represents is adjacent to u and one for which one of the vertices in G_P it represents is adjacent to u'. We place two guards at u and u', all adjacent unmatched vertices representing a matching edge contain vertices within distance at most 2 from u and u'. The two guards cover at least $u, u', w,$ at least one pair of vertices x, x', where $v_{\{x,x'\}}$ unmatched in M' and adjacent to $v_{\{u,u'\}}$ and least another single vertex or vertex pair adjacent to $v_{\{u,u'\}}$. Hence, 2 guards cover at least 6 unit squares—on average, each guard covers at least 3 unit squares.
   (b) The unmatched vertex/vertices are adjacent to v_w: We place a guard at w; it sees u, u', w as well as at least one pair of vertices x, x', where $v_{\{x,x'\}}$ unmatched in M' and adjacent to v_w because all these vertices have distance at most 2 to w. The guard covers at least 5 unit squares.

Each guard covers at least 3 unit squares, hence, we yield that $\lfloor \frac{m}{k+1} \rfloor = \lfloor \frac{m}{3} \rfloor$ guards are always sufficient. □

4 Open Problems

We leave the computational complexity in simple polyominoes and upper bounds on the number of guards necessary to cover polyominoes under k-hop visibility for $k \geq 3$ as open problems.

References

[1] J. O'Rourke, K. Supowit, Some NP-hard polygon decomposition problems, IEEE Trans. Inf. Theory 29 (2) (1983) 181–190.

[2] D. Lee, A. K. Lin, Computational complexity of art gallery problems, IEEE Trans. Inf. Theory 32 (2) (1986) 276–282.

[3] V. Chvátal, A combinatorial theorem in plane geometry, J. Combin. Th. Ser. B 18 (1975) 39–41.

[4] T. Biedl, M. T. Irfan, J. Iwerks, J. Kim, J. S. Mitchell, Guarding polyominoes, in: 27th SoCG, New York, NY, USA, 2011, p. 387–396.

[5] C. Iwamoto, T. Kume, Computational complexity of the r-visibility guard set problem for polyominoes, in: Discrete and Comp. Geom. and Graphs, Springer International Publishing, 2014, pp. 87–95.

[6] M. Garey, D. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman & Co., 1978.

[7] D. E. Knuth, A. Raghunathan, The problem of compatible representatives, SIAM J. Discret. Math. 5 (3) (1992) 422–427.

Watchman Routes under Incidence Visibility Constraint in Convex Polygons

Azadeh Tabatabaei* Mohammad Aletaha† Fardin Shapouri‡ Mohammad Ghodsi§

Abstract

This paper describes a practical version of the Watchman Route Problem for convex polygons. The objective is to plan a continuous closed shortest route inside the polygon from which the entire polygon contour is visible. The route starts from a given point s on the boundary and ends at s. The visibility model which is considered in this work is visibility under incidence constraint τ. A point p on the boundary of the polygon and a point q in the polygon's interior or its boundary are τ-visible if they are mutually visible and the angle between the vector perpendicular to the boundary at p and the vector $\vec{pq}$ is at most equal to a given $\tau \in [0, 90^\circ)$. We present an $O(n^4)$-time algorithm to compute the shortest watchman route where n is the number of vertices of the polygon.

1 Introduction

The Watchman Route Problem is a well-known problem in computational geometry. This problem refers to planning a closed curve in a polygon with a minimum length such that every point on the polygon boundary is visible from at least one point on the curve [3]. Chin and Ntafos [4] proved that for simple polygons the problem is solvable in polynomial time but it is NP-hard for polygons with holes.

There are two general cases for the problem; fixed and floating. In the fixed case, a point s on the boundary is given and the route must start and end at s while in the floating one, there is no given starting point. Considering the fixed case, Chin and Ntafos [4] and Tan et al. [12] proposed $O(n^4)$-time algorithms for simple polygons. For the floating case, Carlsson et al. [2] gave an $O(n^6)$ algorithm which was improved to $O(n^5)$ by Tan [11]. Finally, Dror et al. [5] presented the best-known algorithms with $O(n^3 \log n)$ and $O(n^4 \log n)$ time complexity for the fixed and floating cases, respectively.

Related Works. Heretofore, various versions of the watchman problem are introduced and studied by researchers. The visibility model used in the previously-mentioned works was the general visibility model. In this model, two points are visible to each other if the sight line between them does not intersect the exterior of the polygon. Ntafos introduced watchman routes under limited visibility called d-Watchman Route Problem [9]. In this visibility model, two points are defined to be d-visible if the length of the sight line between them is at most d. Recently, Nilsson et al. considered the rotated monotone visibility model for Watchman Route Problem [8]. In this model, two points are visible to each other if there is a path that connects them such that the path entirely lies in the polygon and it is monotone w.r.t. a given direction $\theta \in [0, 180^\circ)$.

In this paper, we use a visibility model called incidence visibility constraint τ. It was introduced by Gonzalez et al. and used in a variant of the Art Gallery Problem [6]. We will define it in the next section.

Our contribution. We studied the fixed Watchman Route Problem in convex polygons under incidence visibility model with constraint τ (the τ-Watchman Route Problem), this version of the Watchman Route Problem was introduced by Tabatabaei and Mohades [10]. First, we compute a set of points and segments such that every path which covers the polygon contour has to pass through them. Then, using such a set and applying the so-called unfolding method, we compute the τ-Watchman Route. The presented algorithm has $O(n^4)$ time complexity.

As an application of this problem, consider an environment inspection task by a robot-sensor system. If the goal is to process the environment in detail (e.g. watching the images on the walls), the environment contour should be scanned with high clarity. Therefore, not only the existence of the sight line between the robot and the boundary is necessary but also the robot must watch the images from a proper angle.

2 Preliminaries

Throughout this paper, the environment is a convex polygon P having n edges and vertices and its vertices are given in clockwise order. Denote the boundary and interior of the polygon by ∂P and $\imath P$ respectively. Now, we define the visibility model we used as follows.

Definition 1 (Incidence Visibility Constraint τ). Two points p on ∂P and q in P are τ-visible from each other if two following conditions satisfied.

*Department of Computer Engineering, University of Science and Culture, Tehran, Iran, a.tabatabaei@usc.ac.ir

†Department of Computer Engineering, Sharif University of Technology, Tehran, mohammadaletaha@ce.sharif.edu

‡Department of Computer Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran, shapouri@qiau.ac.ir

§Sharif University of Technology and Institute for Research in Fundamental Sciences (IPM), Tehran, Iran, ghodsi@sharif.edu

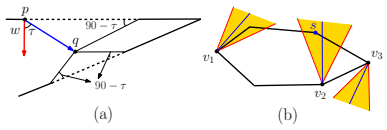


Figure 1: (a) Only the dotted portions of the edges are visible from q. (b) Three types of visibility-regions.

1. Line of sight constraint: The line segment pq does not intersect ∂P except for p and q.

2. Incidence constraint: $\angle(\vec{w}, \vec{pq}) \leq \tau$, where $\vec{pq}$ is a vector directed from p to q, vector $\vec{w}$ is a vector perpendicular to ∂P oriented from p, and τ is a given constant in $[0, 90°)$, see Figure 1(a).

Since the environment in this paper is a convex polygon, the first condition is always true. Note that here, we relax the range constraint defined in [6].

Definition 2 (Fixed τ-Watchman Route Problem). Given a polygon P, the problem asks for a closed route inside P with a minimum length such that each point on ∂P is τ-visible from at least one point on the route. Also, the route must pass through a given vertex s of P [10].

Note that under general visibility model, the fixed watchman route in convex polygons degenerates to the point s. In contrast, under incidence visibility model, the route is not necessarily a point.

Let p be a point on edge e of ∂P. The locus of polygon points (interior and boundary points) from which p is τ-visible is called *visibility-region* of p, denoted by V_p, see Figure 1(b). The visibility-region V_p is dominated by two half-lines called *visibility-rays* denoted by r_1 and r_2. The angle between each visibility-ray and the vector perpendicular to e at p is equal to τ. We classify V_p into three types according to its intersection with P. For a given point p on ∂P, V_p is *internal* if both r_1 and r_2 intersect $\imath P$. If neither r_1 nor r_2 intersect $\imath P$, then V_p is *external*. Otherwise, V_p is *half-internal* (i.e. only one of r_1 and r_2 intersects $\imath P$), see Figure 1(b).

Consider a point p on ∂P such that V_p is external. Clearly, the intersection of V_p and P is equal to p. Hence, p is visible to a point q in P if and only if $q = p$. So, every τ-watchman route must pass through p. Moreover, one can observe that p is a vertex of P and the internal-angle of p is less than $90° - \tau$. We call such a vertex a *critical vertex*. By the above definitions, the following lemma is directly inferred.

Lemma 1 *The τ-Watchman Route must pass through the critical vertices.*

The τ-watchman route must see the whole polygon boundary ∂P. We know that ∂P formed by connecting the edges e_i of P $(1 \leq i \leq n)$. Hence, the τ-watchman route has to see all edges e_i.

Now, consider an edge e with two end-points a and b. We want to plan a continuous path $\mathcal{P}$ such that every point on the segment ab is τ-visible from at least one point on the path. For all points on ab, the union of visibility-regions is dominated by V_a and V_b, see Figure 2(a). It is sufficient for $\mathcal{P}$ to pass through at least one point on both V_a and V_b. Let x and y denote the points where $\mathcal{P}$ intersects V_a and V_b, respectively. The path $\mathcal{P}$ would be each arbitrary path from x to y. Since the path $\mathcal{P}$ is continuous, every point on segment ab is τ-visible from at least one point on $\mathcal{P}$. Consequently, we can state the following lemma.

Lemma 2 *A closed curve is a τ-watchman route if and only if it intersects the visibility-regions of both end-points of each edge of P.*

3 Algorithm

In this section, we describe the algorithm for finding the τ-watchman route in a convex polygon. Briefly, the algorithm starts by computing a set of points and segments which the τ-watchman route has to pass through. We call such a set a *critical set*. Afterward, we obtain two other sets using the critical set. Finally, using the obtained sets and applying an extended version of the so-called unfolding method by Chin and Ntafos [4].

3.1 Obtaining The Critical Set

First, we denote the critical set by C and initialize it to be $\emptyset$. Each edge e_i of P has two end-points v_i and v_{i+1} (in a clock-wise order) which are the vertices of P. So, for each e_i we compute V_{v_i} and $V_{v_{i+1}}$. Every vertex v' is the common end-point of two consecutive edges e'_1 and e'_2 of P. Hence $V_{v'}$ must be computed twice with respect to e'_1 and e'_2.

By Lemma 2, we need to find at least one point in V_{v_i} and $V_{v_{i+1}}$. As we consider the fixed version of τ-watchman route, the route must pass through a given vertex s of P. So, we already know s is on the route. Considering point s, for each end-point v_1 of an edge e_1 we have four cases:

Case 1: *If V_{v_1} is internal and does not include s.*

The τ-watchman route needs to visit only a point in V_{v_1}. Since V_{v_1} is internal, two visibility-rays of v_1 intersect $\imath P$. We choose the one which is closest to s and denote it by r_0. Then, using the ray-shooting operation, we compute a segment obtained by the intersection of ray r_0 and P. We call such a segment a *critical segment* and we add it to the critical set C.

Case 2: *If V_{v_1} is half-internal and does not include s.*

In this case, only one visibility-ray of v_1 intersects $\imath P$. Denote that visibility-ray by r_0. Similar to case 1, we compute the segment obtained from the intersection of r_0 and P. This segment is also called a *critical segment* and we add it to C.

Case 3: *If V_{v_1} is internal/half-internal and includes s.*

Since the τ-watchman route pass through s and s sees v_1, the τ-watchman route already sees v_1. Thus, there is no change in C.

Case 4: *If V_{v_1} is external.*

The vertex v_1 is a critical vertex. By Lemma 1, the τ-watchman route must pass through v_1. Hence, we add v_1 in C.

The critical set C can be obtained by doing the above case analysis for the visibility-regions of end-points of all edges. To computing the visibility regions, first, we triangulate P. Since P is convex, we pick an arbitrary vertex and put in the diagonals to other vertices. It takes linear time for triangulation. Then, we use the ray-shooting data structure for triangulated polygons by Guibas et al. [7]. Each ray-shooting operation takes $O(\log n)$ time. Knowing the end-points of a segment obtained by the intersection of a visibility-ray and P, we can determine the position of s concerning the visibility-region in $O(1)$ time. Therefore, the overall time taken for computing C is $O(n \log n)$.

Consider a ray r which has two intersection points l_1, l_2 with ∂P. If c is the critical segment obtained from r, it can be represented by its end-points, e.g. $c = l_1 l_2$. Also, one can observe that each critical vertex v is a critical segment which degenerated to a point. In other words, critical vertex v has two intersection points l_1, l_2 with ∂P such that $v = l_1 = l_2$. Hereinafter, we call them critical segment as well.

3.2 Obtaining The Essential Set

Each segment c partitions P into two regions; one of which include s and another which does not. We denote the region including s by *essential region* of c.

Let $c = l_1 l_2$ and $c' = l'_1 l'_2$ be two critical segments in C. Assume c and c' do not cross each other in $\imath P$. In this case, w.l.o.g. assume that if $l_1 l_2$ partitions P into two regions, then $l'_1 l'_2$ falls in the region which does not include s. This means if we traverse ∂P in the clock-wise order from s, the points l'_1, l'_2 will appear between the appearances of l_1 and l_2. In this case, every path from s to a point on c' should cross c. We call c *dominated* by c', see Figure 2(b).

Note that critical vertices (degenerated critical segments) dominate some critical segments but they can not be dominated by any critical segments.

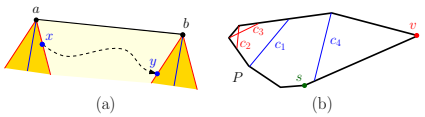


Figure 2: (a) Segment ab is visible from the path from x to y (b) Vertex v dominates segment c_4, and c_1 is dominated by c_2 and c_3.

Each element of C which are not dominated by any other element is more important than others. We call such a critical segment an *essential segment*. It is sufficient for the τ-watchman route to pass through essential segments. So, other critical segments can be ignored.

The set of essential segments is called the *essential set* and denoted by E. The set E can be obtained from C by removing the dominated elements of C. We obtain E in a similar way to the algorithm for obtaining essential cuts in [2]. We know $|C| = O(n)$. So, an essential segment c_0 can be found in $O(n)$. Then, using c_0, by traversing all elements of C, the set E can be obtained. This process is explained in [2]. It takes $O(n)$ time for traversing the elements of C.

3.3 Obtaining The Fragment Set

As we observe, in E some of essential segments might cross each other, see Figure 2(b). A segment induced by intersection of essential segments called a *fragment*. Given the obtained set E, the set of fragments F can be computed by the algorithm for line segment intersections [1]. Since $|E| \leq |C| = O(n)$ and $|F| \leq |E|^2$, computing fragments takes $O(|E| \log |E| + |F|) = O(n^2)$.

3.4 The Extended Unfolding Method

Heretofore, we obtained a set of essential segments E and a set of fragments F. Having similar sets E' and F' under general visibility model, the algorithm by Chin and Ntafos can compute the shortest fixed watchman route. First, we claim it as the following theorem.

Theorem 3 *v Given a simple polygon P and the set of essential cuts E' and subdivision of essential cuts into fragments F', there is an algorithm which computes the fixed shortest watchman route under general visibility model in $O(n|E'||F'|)$ time.*

Here, the sets E and F are computed under incidence visibility model. Also, convex polygons are already simple. Hence, we can give the sets E and F as an input to the above-mentioned algorithm. The resulting route is the fixed shortest τ-watchman route.

We briefly describe the algorithm by Chin and Ntafos [4]. As we discussed earlier in this section, the τ-watchman route must have at least a point on each

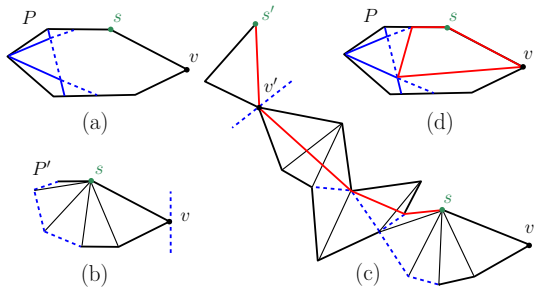


Figure 3: Illustrating the extended unfolding method.

essential segment. Also, the fragments are made by dividing essential segments at their intersection points. One can observe that the τ-watchman route does not need to touch all of the fragments, but a subset of fragments is sufficient to be touched by the route. Such a subset is called the subset of *active-fragments*. An essential segment which contains an active-fragment is also called an *active-segment*.

The main idea of the algorithm is computing an optimum watchman route by the unfolding method and repeatedly applying some adjustments to the current route to make the route shorter. The adjustments are applied for selecting the set of active-segments. In the end, the resulting route is the shortest watchman route.

Now, we describe the *unfolding method* used by the above algorithm. Given a set of active-segments, first remove non-essential regions of the active-segments from P to have a reduced polygon P', see Figure 3(b). Then, P' is triangulated. Starting from the point s in a clockwise order, when we reach the first active-segment c_1, we take it as a mirror and construct a new polygon by reflecting the triangles traversed from s to c_1. Then, we attach this polygon to the previous polygon. Now, we traverse from c_1 to second active-segment c_2 and we repeat this process for all active-segments. At the end of traversal, we again back to s, and the polygon P' is unfolded using active-segments as mirrors, see Figure 3(c). After that, we find the shortest path from s to its image s' in the unfolded polygon. Finally, the path from s to s' is folded back and yields a watchman route for the given active-segments, see Figure 3(d).

While the unfolding method only reflects on segments, we extend it to also reflect on critical vertices that appear in the set of active-segments. In fact, we reflect on an external tangent-line to P' at each critical vertex, see Figure 3(b). By theorem 3, applying the above-mentioned algorithm takes $O(n|E||F|) = O(n^4)$ time since $|E| = O(n)$ and $|F| \leq |E|^2$. The time complexity of the last step of the algorithm dominates the other steps (obtaining the sets C, E, and F). As a consequence, we can conclude the following Theorem.

Theorem 4 *Given a convex polygon P and a vertex s on P the fixed shortest τ-watchman route can be computed in $O(n^4)$ time.*

4 Conclusion

In this paper, we presented an $O(n^4)$-time algorithm for the fixed watchman route problem under incidence visibility constraint τ in convex polygons. It is interesting to consider the floating version of the problem. Also, developing algorithms for the watchman route problem under incidence visibility constraint τ in other polygonal environments remained as a challenging open problem.

References

[1] I. J. Balaban. An optimal algorithm for finding segments intersections. In *Proceedings of the eleventh annual symposium on Computational geometry*, pages 211–219, 1995.

[2] S. Carlsson, H. Jonsson, and B. J. Nilsson. Finding the shortest watchman route in a simple polygon. *Discrete & Computational Geometry*, 22(3):377–402, 1999.

[3] W. Chin and S. Ntafos. Optimum watchman routes. In *Proceedings of the second annual symposium on Computational geometry*, pages 24–33, 1986.

[4] W. Chin and S. Ntafos. Shortest watchman routes in simple polygons. *Discrete & Computational Geometry*, 6(1):9–31, 1991.

[5] M. Dror, A. Efrat, A. Lubiw, and J. S. Mitchell. Touring a sequence of polygons. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 473–482, 2003.

[6] H. González-Baños. A randomized art-gallery algorithm for sensor placement. In *Proceedings of the seventeenth annual symposium on Computational geometry*, pages 232–240, 2001.

[7] L. Guibas, J. Hershberger, D. Leven, M. Sharir, and R. E. Tarjan. Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons. *Algorithmica*, 2(1-4):209–233, 1987.

[8] B. J. Nilsson, D. Orden, L. Palios, C. Seara, and P. Żyliński. Shortest watchman tours in simple polygons under rotated monotone visibility. In *International Computing and Combinatorics Conference*, pages 311–323. Springer, 2020.

[9] S. Ntafos. Watchman routes under limited visibility. *Computational Geometry*, 1(3):149–170, 1992.

[10] A. Tabatabei and A. Mohades. Clarity watchman route. *7th Japan Conference on Computational Geometry and Graphs,* JCCGG 2009.

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

[12] X. Tan, T. Hirata, and Y. Inagaki. Corrigendum to "an incremental algorithm for constructing shortest watchman routes". *International Journal of Computational Geometry and Applications*, 9(03):319–323, 1999.

Simple Robot Free-Target Search in Rectilinear Streets

Mohammad Aletaha* Arash Vaezi* Mohammad Abouei Mehrizi† Mohammad Ghodsi‡

Abstract

We consider the problem of searching for an arbitrary (random) target in an unknown rectilinear-street using a simple robot. A simple polygon with respect to two given vertices u, v on the boundary is a street if the clockwise and counter-clockwise chains from u to v are weakly visible from each other. A simple robot, also called gap-detector, can only detect the discontinuities in the depth information (gaps) in a cyclical order. The goal is to design a strategy for a simple robot to find an arbitrary target t in a rectilinear-street, starting from u or v, with a minimum path length. We propose a strategy that guarantees a path which is at most $\sqrt{10}$ times longer than the shortest path.

1 Introduction

Path planning for robots in unknown environments is one of the fundamental problems in the fields of computational geometry, online algorithms, and robotics [4]. In path planning problems, a robot must find a target in a specified environment. Note that if we have the geometric map of the environment and the position of the target point, we can find the shortest-path easily. However, we consider the case that the robot does not have the information in advance. Such a robot should follow an online algorithm to find the target. A simple robot has access only to its local information about its surroundings. We denote the start and target point of the robot by s and t, respectively. The *competitive ratio* is the length of the path traveled by the robot from s to t, over the length of the shortest-path. A strategy is called *c-competitive* if its competitive ratio is at most c.

A *street* is a simple polygon with two distinct vertices u and v so that clockwise and counter-clockwise chains from u to v (L_{chain} and R_{chain} resp.) are mutually weakly visible [6]. In other words, every point on each chain is visible to at least one point on another one. A rectilinear-street is an orthogonal street (see Figure 1).

Using simple robots has quite a few advantages over 360° vision robots such as low cost, less sensitivity to failure, easy to replace and maintenance. Therefore, numerous types of simple robots have been characterized and deployed in path planning problems. [8, 12]

We use a simple point robot called *gap-detector*. To understand what a gap-detector robot can do we need to determine a sensor called a *gap-sensor*. A gap-sensor is a minimal sensing model that was introduced by Tovar et al. [12]. It can only detect the discontinuities in the depth information (gaps) in the robot's visibility region and reports them in a cyclical order. It can assign a label L (left-gap) or R (right-gap) to each gap depend on the portion of the environment hidden behind that gap (see Figure 1(a)). Tovar et al. proposed a data structure, called Gap Navigation Tree (GNT), to maintain and update the gaps that have been seen during the robot's movement. Also, the robot can detect the target whenever the target enters its visibility polygon. Later, this robot gets empowered by a 4-wind compass sensor [11]. This latter sensor can illustrate the four main directions (cardinal points: North, East, South, and West), and it can report which gap is between which two main directions or if it is collinear with a main direction.

A simple robot with those characteristics can only move toward the gaps, the compass directions, and the target (when it becomes visible). The robot can move around the polygon through an arbitrary number of steps. A step is a fixed distance specified by the robot's manufacturer. We assume the step is small enough compared to the scale of the given polygon. For the sake of simplicity, we assume that the given rectilinear polygon is based on a grid with unit distance d_g, and the robot's step is equal to d_g/k for any $k \in \mathbb{N}$.

As the robot moves, the combinatorial structure of its visibility region changes by the occurrence of four *critical events*. These critical events are: appearance, disappearance, split, and merge of gaps [12]. An appearance/disappearance event occur when the robot crosses an inflection-ray of a gap. Also, a split/merge event occur when the robot crosses the bitangent-complement of two polygon's reflex vertices (see Figure 2(a)). When a gap appears and the portion behind it was so far visible, we call it a *primitive-gap*. All other gaps are called *non-primitive-gaps*. The robot stores all of these information in GNT.

Previous Works. In 1992, Klein introduced street polygons [6]. He considered the problem of searching in a street, starting from u or v, for the other one. He

*Department of Computer Engineering, Sharif University of Technology, Tehran, Iran, mohammadaletaha@ce.sharif.edu, avaezi@ce.sharif.edu

†Gran Sasso Science Institute (GSSI), L'Aquila, Italy, mohammad.aboueimehrizi@gssi.it.

‡Sharif University of Technology and Institute for Research in Fundamental Sciences (IPM), Tehran, Iran, ghodsi@sharif.edu

presented a 5.73-competitive strategy and proved that the lower bound on the competitive ratio is $\sqrt{2}$. After several improvements, finally, Schuierer and Semrau [7] and Icking et al. [5] independently presented optimal strategies. The robot used in all previously mentioned works equipped with a 360° vision system. Such a robot can detect edges and vertices, measure the distances and angles, and move freely in any direction.

Bröcker and López-Ortiz considered two new types of search in streets called Position-Independent search [2]. In the first type, the robot starts from u (or v) and searches for an arbitrary target t on the boundary. In the second type, both of the start and target points are arbitrary points on the boundary. They presented 36.8 and 69.2-competitive strategies and proved the lower bounds of 9 and 11.78 for these two types, respectively. Bröcker and Schuierer showed that for the rectilinear-streets, one can achieve better competitive ratios [3]. In the first type, they presented an optimal 2.61-competitive strategy and proved a matching lower bound. In the second type, a 59.91-competitive strategy in L_1-metric is proposed.

For the first time, Tabatabaei and Ghodsi [10] considered the gap-detector robots for the Klein's problem [6]. They equipped the robot with a tool called pebble. A pebble is a detectable object that the robot can carry and put it everywhere in the polygon. They have presented an 11-competitive strategy using one pebble. They proved the competitive ratio can be improved to 9 using enough pebbles. Moreover, they showed considering rectilinear-streets, the gap-detector robot which is empowered by a compass achieves the optimal competitive ratio of $\sqrt{2}$. Wei et al. [13] and Tabatabaei et al. [9] independently presented 9-competitive strategies without using any pebbles. Furthermore, in [9] a 7-competitive randomized strategy has been proposed. Additionally, a lower bound of 9 (4.59) on the competitive ratio of all deterministic (randomized) strategies has been proved [9, 13]. Recently, Tabatabaei et al. showed that empowering the robot with a compass improves the competitive ratio to $3\sqrt{2}$ [11]. It also has been shown that if two simple robots cooperate with each other, the competitive ratio will decrease to 2 [1].

Our Contribution. We consider the first type of the Position-Independent search (mentioned above) introduced in [2] and call it *Free-Target search*. Inspiring from [3] in which the authors considered a 360° vision robot, we present a $\sqrt{10}$-competitive strategy using a gap-detector robot. Please note that despite 360° vision robots, gap-detector robot's vision and movement are strictly limited.

Problem Definition. Given a rectilinear-street polygon $\mathcal{P}$ (with respect to two vertices u, v), a simple robot $\mathcal{R}$ standing on a start vertex $s \in \{u, v\}$. Is there an online strategy with the minimum path for $\mathcal{R}$ to move from s and find a given target point t in $\mathcal{P}$? In the rest of the paper, we call it Free-Target Search (FTS) problem.

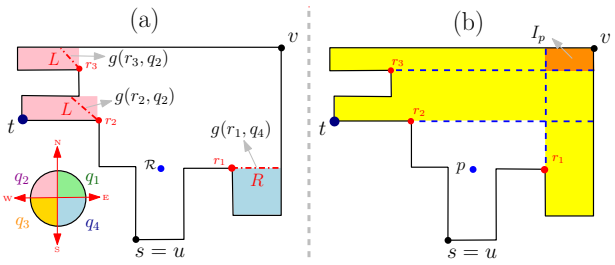


Figure 1: (a) A rectilinear-street, the robot $\mathcal{R}$, and the gaps (dashed lines). (b) Unexplored-regions of gaps at point p and their intersection area I_p.

2 Strategy

This section presents an online strategy for the FTS problem. Without loss of generality, assume that $s = u$. A robot $\mathcal{R}$ needs to explore $\mathcal{P}$ to find t. The target t may lie behind any gap. If the area behind a gap g is already visited by $\mathcal{R}$, the gap is primitive, the robot no longer needs to explore g. Hence, only non-primitive gaps are required to get visited. Hereinafter, when we use *gap*, we mean a non-primitive gap. As we mentioned earlier, the gap-detector robot has a compass sensor with four main directions. Each gap either falls between two main directions or is collinear with a main direction. Main directions partition the environment to four quadrants: NE, SE, SW, and NW. We assign each gap which falls between two directions to the related quadrant. Consider a gap g which is collinear with a main direction d, one side of d is visible to $\mathcal{R}$ and the other side is invisible. For a gap g which is collinear with d, we assign it to the quadrant contains the hidden part of g.

The strategy has three cases. Each case has an initiation-point and an end-point. A procedure called *case-analysis* determines which case the robot $\mathcal{R}$ should choose for its next step. From the beginning point (s), $\mathcal{R}$ should run case-analysis. At the end-point of each case, the robot $\mathcal{R}$ stops moving and runs case-analysis. Note that an end-point of a case is an initiation-point of the next case.

Whenever t is visible by $\mathcal{R}$, it stops and moves directly toward t regardless of its previous direction. Based on the position of $\mathcal{R}$ in $\mathcal{P}$, and the circumstances of the gaps around $\mathcal{R}$, there are three cases mentioned in the following. For a better presentation, based on the position of $\mathcal{R}$, each gap g is denoted by $g(r, q)$ which r is a reflex vertex that causes $\mathcal{R}$ not to see a part of $\mathcal{P}$, and q is a quadrant that g is assigned to it (see Figure 1(a)).

A case ends (or a new case initiates) when a gap ap-

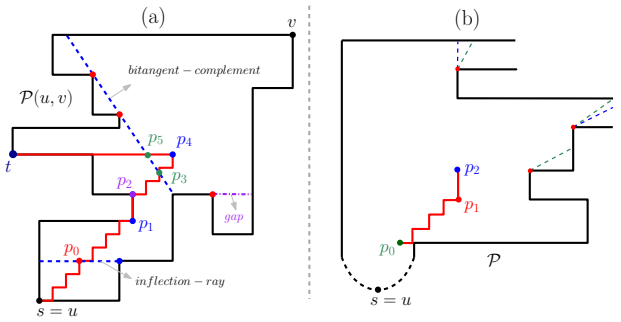
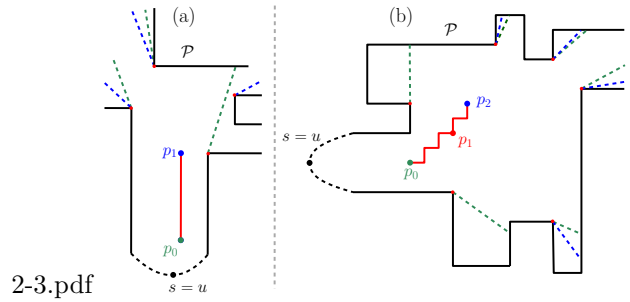


Figure 2: (a) Examples of critical events: at points p_0, p_2, p_3, p_5, the events disappearance, appearance, split, and merge occur, respectively. (b) Case 1.



2-3.pdf

Figure 3: (a) Case 2 (b) Case 3.

pears or disappears. The robot $\mathcal{R}$ stops when a case is finished and run the case-analysis procedure. That is because the proposed strategy works based on the positions of gaps around $\mathcal{R}$. In fact, the number of quadrants the gaps are assigned to determines the strategy cases. So, an appearance or disappearance event might change the current case of $\mathcal{R}$.

Furthermore, note that the below-mentioned strategy guarantees that a merge or a split event can never change the current case of $\mathcal{R}$. That is because when either a merge or split event occurs for more than one gap, all such gaps must remain in the same quadrant. So, the current case will not change. Consequently, the robot considers those events only for updating GNT, but ignores them for case-analysis; in fact, only appearance and disappearance events will be considered for case-analysis.

Case 1: *If gap(s) is (are) assigned to one quadrant q (see Figure 2(b)).*
The robot moves alternatively toward two directions adjacent to q, i.e., one step toward a direction and one step toward another one. The robot $\mathcal{R}$ stops if a gap becomes collinear with any of its two main directions, say d, then $\mathcal{R}$ turns and moves directly toward d (see point p_1 in Figure 2(b)).

Case 2: *If gaps are assigned to two adjacent quadrants q and q' (see Figure 3(a)).*
The robot $\mathcal{R}$ moves along the main direction between q and q', e.g., if the quadrants be NE and NW, the robot moves toward N.

Case 3: *If gaps $g_i(r_i, q)$, $g_j(r_j, q')$, and $g_k(r_k, q'')$, $1 \leq i, k$ exist and at least some of them are located in two non-adjacent quadrants (q, q'') (see Figure 3(b)).*
In this case, the gaps are assigned to at most three quadrants. There are at least two opposite side quadrants q, q''. We call each gap which is assigned to q or q'' a *crucial-gap*.

If $j \geq 1$ (at least one gap is assigned to q'), then $\mathcal{R}$ moves alternatively between two main directions of q' (see Figure 3(b)).

Else there is no middle quadrant, and the opposite quadrants q and q'' must be either q_1, q_3 or q_2, q_4 (see Figure 1(a)). Using the compass, $\mathcal{R}$ can distinguish between these two. In such a case we need to find a quadrant to be in the placed of q'. This is a quadrant where it is not visited yet, and $\mathcal{R}$ should move towards q'. We set q' to be that quadrant, then $\mathcal{R}$ should move alternatively between two main directions of q'. In the following we will see how to find q':

If $q, q'' == q_1, q_3$, then choose an arbitrary $g_i(r_i, q)$ and see if g_i is a right-gap then q' equals to q_2. Otherwise, if g_i is a left-gap then q' equals to q_4.

Else if $q, q'' == q_2, q_4$, then choose an arbitrary $g_i(r_i, q)$ and see if g_i is a right-gap then q' equals to q_3. Otherwise, if g_i is a left-gap then q' equals to q_1.

3 Analysis

This section covers the proof of correctness and competitive ratio of FTS strategy mentioned in section 2. For every gap $g(r, q)$, r is a reflex vertex adjacent with two edges of $\mathcal{P}$, one of these two edges is visible to $\mathcal{R}$, and the other one is hidden. The extension of the hidden edge into the interior of $\mathcal{P}$ is called *inflection-ray* of r [10]. When $\mathcal{R}$ crosses the inflection-ray of r, $g(r, q)$ disappears accordingly. Every inflection-ray partitions $\mathcal{P}$ into two regions, one includes u and the other one contains v [3]. The latter one is called *unexplored-region* (see Figure 1(b)). Consider a point p as the current position of $\mathcal{R}$. We denote the intersection area of all unexplored-regions of the gaps at point p as I_p. Appendix A covers all the lemmas and observations that we used in the proof.

Theorem 1 *Using the strategy presented in Section 2, $\mathcal{R}$ always find t.*

Proof. The robot $\mathcal{R}$ always faces one of the three cases defined in Section 2 (based on Lemma 4 in Appendix A).

Consider the point t' from which $\mathcal{R}$ sees t. Thus it is sufficient to prove that $\mathcal{R}$ always directs to I_p for $p \in path(s, t')$, regarding those three cases. Let investigate each case as follows.

- Case 1: All of the gaps are located in a quadrant q, and $\mathcal{R}$ moves along an L_1-path through q. In fact, $\mathcal{R}$ must cross the inflection-ray of a gap. Since I_p is behind the unexplored-regions and $\mathcal{R}$ goes toward I_p, then $\mathcal{R}$ must cross an inflection-ray.

- Case 2: By the strategy, $\mathcal{R}$ moves toward a main direction d until an appearance/disappearance event happens. So similar to Case 1, $\mathcal{R}$ must cross an inflection-ray of a gap. In fact, at least one gap must get disappeared when $\mathcal{R}$ moves toward d and $\mathcal{R}$ reaches the inflection-ray of the disappeared gap. Since I_p (for any $p \in path(s, t')$) is behind all of the unexplored-regions, $\mathcal{R}$ must be directed to I_p. When $\mathcal{R}$ moves toward d, there must be a gap that get disappeared, otherwise $\mathcal{P}$ is not street. We claim that there are at least one gap whose inflection-ray is perpendicular to d. By contradiction, assume all inflection-rays of all gaps are collinear with d. Then, the intersection of the unexplored-regions of gaps must be null, and as a result $I_p = \emptyset$, which is not possible. So, there must be a gap that got disappeared while $\mathcal{R}$ moves toward d (see Figure 4(b)).

- Case 3: In this case, I_p (for any $p \in path(s, t')$) lies in the middle quadrant q'. The strategy leads $\mathcal{R}$ to move alternatively along two main directions adjacent to q' (the unexplored middle quadrant). So, $\mathcal{R}$ moves along an L_1-path toward I_p.

In all cases, we showed that $\mathcal{R}$ always pointed to I_p ($p \in path(s, t')$) and never got away from it. As a result, $\mathcal{R}$ must finally find t, and this concludes the proof. □

The following theorem demonstrates that the competitive ratio of the strategy is $\sqrt{10}$. Moreover, Theorem 3 demonstrates that when $t = v$ the competitive ratio is $\sqrt{2}$. Again, Appendix B covers their proofs.

Theorem 2 *Given a rectilinear-street polygon $\mathcal{P}$ (u, v), and a simple robot $\mathcal{R}$, the strategy presented in Section 2 is $\sqrt{10}$-competitive.*

Theorem 3 *If $t = v$, the competitive ratio is $\sqrt{2}$ and it is optimal.*

4 Conclusion

We studied the problem of Free-Target Search in an unknown rectilinear-street for a simple robot. The robot starts from one of two distinguished points u or v and searches for an arbitrary target point t. We used a gap-detector robot that has a minimal sensing capability. Our strategy generates a path, starts at u (or v) to t, with a competitive ratio of $\sqrt{10}$. This paper opens several research lines. We plan to consider the problem when the robot starts from an arbitrary point on the boundary. Another research line is to study the problem in more general environments like streets and generalized-streets.

References

[1] M. Abouei Mehrizi, M. Ghodsi, and A. Tabatabaei. Robots' cooperation for finding a target in streets. In *International Conference on Topics in Theoretical Computer Science*, pages 30–43. Springer, 2015.

[2] C. A. Bröcker and A. López-Ortiz. Position-independent street searching. In *Workshop on Algorithms and Data Structures*, pages 241–252. Springer, 1999.

[3] C. A. Bröcker and S. Schuierer. Searching rectilinear streets completely. In *Workshop on Algorithms and Data Structures*, pages 98–109. Springer, 1999.

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

[5] C. Icking, R. Klein, and E. Langetepe. An optimal competitive strategy for walking in streets. In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 110–120. Springer, 1999.

[6] R. Klein. Walking an unknown street with bounded detour. *Computational Geometry*, 1(6):325–351, 1992.

[7] S. Schuierer and I. Semrau. An optimal strategy for searching in unknown streets. In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 121–131. Springer, 1999.

[8] S. Suri, E. Vicari, and P. Widmayer. Simple robots with minimal sensing: From local visibility to global geometry. *The International Journal of Robotics Research*, 27(9):1055–1067, 2008.

[9] A. Tabatabaei, M. Aletaha, and M. Ghodsi. Competitive strategies for walking in streets for a simple robot using local information. In *The Third Iranian Conference on Computational Geometry*, pages 15–19, 2020.

[10] A. Tabatabaei and M. Ghodsi. Walking in streets with minimal sensing. *Journal of Combinatorial Optimization*, 30(2):387–401, 2015.

[11] A. Tabatabaei, M. Ghodsi, and F. Shapouri. Competitive strategy for walking in streets for an empowered simple robot. ICCG, 2019.

[12] B. Tovar, R. Murrieta-Cid, and S. M. LaValle. Distance-optimal navigation in an unknown environment without sensing distances. *IEEE Transactions on Robotics*, 23(3):506–518, 2007.

[13] Q. Wei, X. Tan, and Y. Ren. Walking an unknown street with limited sensing. *International Journal of Pattern Recognition and Artificial Intelligence*, 33(13):1959042, 2019.

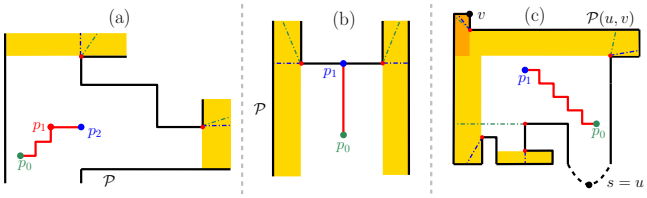


Figure 4: Examples of forbidden cases in which $\mathcal{R}$ sees some gaps at p but $I_p = \emptyset$. These cases does not occur through the robot's path in a street. If such a cases occur the polygon is not a street.

Appendix

A Correctness

This section covers the proof of correctness of the FTS strategy mentioned in section 2. For every gap $g(r,q)$, r is a reflex vertex adjacent with two edges of $\mathcal{P}$, one of these two edges is visible to $\mathcal{R}$, and the other one is hidden. The extension of the hidden edge into the interior of the polygon is called *inflection-ray* of r [10]. When $\mathcal{R}$ crosses the inflection-ray of r, $g(r,q)$ disappears accordingly. Every inflection-ray partitions the polygon into two regions, one includes u and the other one contains v [3].The latter one is called *unexplored-region* (see Figure 1(b)). Consider a point p as the current position of $\mathcal{R}$. We denote the intersection area of all unexplored-regions of the gaps at point p as I_p. In this section, we consider proving the following theorem.

Theorem 1: Using the strategy presented in section 2, $\mathcal{R}$ always find t.

Before proving the statement, let demonstrate some preliminaries.

Observation 1 *If $\mathcal{R}$ sees at least one gap at p, the point v must lie in I_p.*

Proof. By contradiction, let assume that v is in the unexplored-region of a gap $g(r_0,q)$ but not in I_p. As a result, the intersection of the boundary of $\mathcal{P}$ and I_p does not contain v. Also, we already know that I_p does not contain u. If a part of the boundary of $\mathcal{P}$ does not contains neither u nor v, then it must belongs to either one of L_{chain} or R_{chain} of $\mathcal{P}$. Consider a reflex vertex r_1 whose unexplored-region contains I_p but not v. As a result, the points on the hidden-edge of r_1 is not visible from the other chain, which contradicts the definition of a street polygon. □

Note that if there is at least one non-primitive gap, since I_p always contains at least v, $I_p \neq \emptyset$. So, the cases illustrated in Figure 4 can never occur.

Remark 2 *Since the polygon is rectilinear, for every gap $g(r,q)$, the unexplored-region of g can lie in q and at most one of its adjacent quadrants, but it never lies in the opposite (non-adjacent) quadrant of q.*

Observation 3 *Consider p as the current position of $\mathcal{R}$, if there exist gaps in two opposite quadrants (q, q''), I_p must be in the unexplored middle quadrant (q').*

Proof. Since $I_p \neq \emptyset$, the unexplored-regions of the gaps $g(r,q), g(r'',q'')$ must have an intersection region. The unexplored-regions of two opposite quadrants can only intersect in an unexplored quadrant named middle quadrant and denoted by q'. That is because their common region cannot be in q or q'' and must be somewhere in q' (see Figure 1(b)). □

In the next lemma, we show that regardless of the robot's position, the gaps can only lie in at most three quadrants.

Lemma 4 *At each point p of the robot's path, all the gaps must be located in at most three quadrants.*

Proof. Assume a situation where there are some gaps in three quadrants; w.l.o.g. assume that they are located in NE, NW, and SE. Hence, according to the Observation 3, I_p will be located in the middle quadrant, i.e., NW. By contradiction, assume there exists a gap $g(r,SW)$. As we saw in Remark 2, the unexplored-region of $g(r,SW)$ could be located in any of the quadrants except NW. Consequently, I_p is empty ($I_p = \emptyset$) and contradicts Observation 1 (see Figure 4(c)). □

Denote a path generated by the strategy from x to y by $path(x,y)$. We show that the strategy generates a path from $s=u$ to t ($path(s,t)$). Consider the point t' from which $\mathcal{R}$ sees t. It is sufficient to show that the strategy generates a path from s toward I_p for any $p \in path(s,t')$.

At each point $p \in path(s,t')$ if there exist at least one gap, we know v is in I_p. So, $\mathcal{R}$ travels a path from s toward v. Hence, either the whole polygon is cleared before reaching v, or v is reached by $\mathcal{R}$. At the latter case, $\mathcal{R}$ traveled a $path(s,v)$. Klien [6] proved that every path from $s=u$ to v explores the whole polygon. Therefore, at both cases, the whole polygon gets explored and $\mathcal{R}$ surely sees t.

Consider a point p_f on the boundary of last unexplored I_p $p \in path(s,t')$, when $\mathcal{R}$ reaches I_{p_f} all the gaps must disappear, and the whole polygon is clear. That is because v lies in I_{p_f}. So, the robot already passed over the path $path(s,p_f)$, and lies on p_f, and the only unexplored region of $\mathcal{P}$ (I_{p_f}) gets visible for $\mathcal{R}$. For every $p' \in path(p_f,t)$ $I_{p'} = \emptyset$. So, we only need to show that $\mathcal{R}$ always moves toward I_p for any $p \in path(s,t')$. We will show that considering the strategy's cases, the robot can eliminate each non-primitive-gap only once. Hence, the total path of $\mathcal{R}$ is finite, and the strategy will terminate.

Proof. [Proof of Theorem 1] Based on Lemma 4, $\mathcal{R}$ always faces one of the three cases defined in Section 2. Thus it is sufficient to prove that $\mathcal{R}$ always directs to I_p for any $p \in path(s,t')$, regarding all those three cases. We investigate each case separately, as follows.

- Case 1: All of the gaps are located in a quadrant q, and $\mathcal{R}$ moves along an L_1-path through q. In fact, $\mathcal{R}$ must cross the inflection-ray of a gap. Since I_p is behind the unexplored-regions and $\mathcal{R}$ goes toward I_p, then $\mathcal{R}$ must cross an inflection-ray.
- Case 2: By the strategy, $\mathcal{R}$ moves toward a main direction d until an appearance/disappearance event happens.

So similar to Case 1, $\mathcal{R}$ must cross an inflection-ray of a gap. In fact, at least one gap must get disappeared when $\mathcal{R}$ moves toward d and $\mathcal{R}$ reaches the inflection-ray of the disappeared gap. Since I_p (for any $p \in path(s,t')$) is behind all of the unexplored-regions, $\mathcal{R}$ must be directed to I_p. When $\mathcal{R}$ moves toward d, there must be a gap that get disappeared, otherwise $\mathcal{P}$ is not street. We claim that there are at least one gap whose inflection-ray is perpendicular to d. By contradiction, assume all inflection-rays of all gaps are collinear with d. Then, the intersection of the unexplored-regions of gaps must be null, and as a result $I_p = \emptyset$, which is not possible. So, there must be a gap that got disappeared while $\mathcal{R}$ moves toward d (see Figure 4(b)).

- Case 3: In this case, I_p (for any $p \in path(s,t')$) lies in the middle quadrant q'. The strategy leads $\mathcal{R}$ to move alternatively along two main directions adjacent to q' (the unexplored middle quadrant). So, $\mathcal{R}$ moves along an L_1-path toward I_p.

In all cases, we showed that $\mathcal{R}$ always pointed to I_p (p is a point in $path(s,t')$) and never got away from it. As a result, $\mathcal{R}$ must finally find t, and this concludes the proof. □

B Competitive Ratio

In this section, we intend to prove that the strategy stated in section 2 provides a competitive ratio of $\sqrt{10}$ for a simple robot $\mathcal{R}$.

During the robot's movement, whenever t becomes visible, $\mathcal{R}$ can detect it and move toward it. Let denote the last intersection of all unexplored-regions that $\mathcal{R}$ may see in its path ($path(s,t)$) by I_{p_f}, where p_f is a point on the boundary of last unexplored I_p (for any $p \in path(s,t')$).

Consider a situation where $\mathcal{R}$ lies on p_f. In such a case, t is located in I_{p_f} and $\mathcal{R}$ moves directly toward t. Note that if t is not in I_{p_f}, $\mathcal{R}$ must have seen it before reaching p_f.

In the following theorem, we show that all cases provide an L_1-shortest-path but Case 3. In fact, the competitive ratio for the first two cases is $\sqrt{2}$, while it is $\sqrt{10}$ for case 3.

Theorem 2: Given a rectilinear-street polygon $\mathcal{P}$ (u,v), and a simple robot $\mathcal{R}$, the strategy presented in Section 2 is $\sqrt{10}$-competitive.

Proof. When t is not visible to $\mathcal{R}$, it is hidden behind a gap $g(r,q)$. Consider a situation where t is located on the line containing an inflection-ray. Suppose $\mathcal{R}$ is not reached that inflection-ray yet, and the target point t is on the hidden edge $g(r,q)$. In this situation, $\mathcal{R}$ cannot see t until it reaches the inflection-ray of r. Regarding the positions of t and r, in other situations $\mathcal{R}$ might see t before it reaches the inflection-ray of r, and it will pass a shorter path to meet t (see Figure 5(a)). Hence, when t is on the hidden edge of a gap, $\mathcal{R}$ has to pass longer path concerning any other position of t.

For the first two cases, we show that when $\mathcal{R}$ reaches t, the robot's path length is at most equal to an L_1-shortest-path from s to t. This is because $\mathcal{R}$ approaches to quadrant(s) in which there exist gap(s). Note that R never gets away from gaps. We consider each case separately in the following.

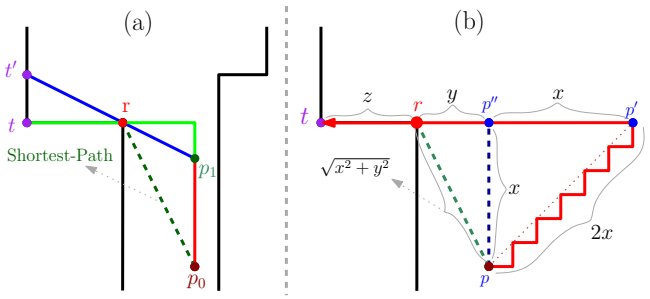


Figure 5: Worst case for finding t.

- Case 1: In this case, t is behind one of the gaps. Since all of the gaps are located in the same quadrant (q), and $\mathcal{R}$ moves toward q (one step in each direction alternatively), then its path length is equal to the L_1-shortest-path (see Figure 2(b)).
- Case 2: In this case, the gaps are in two adjacent quadrants q, q', and $\mathcal{R}$ moves toward the main direction between them. Since t is behind one of the gaps when R goes straight between q, q', it guarantees an L_1-path as $\mathcal{R}$ will never go away from t, and then its length is equal to the L_1-shortest-path (see Figure 3(a)).

In the above cases, $\mathcal{R}$ passes through an xy-monotone path, and the length of the path is equal to an L_1-shortest-path. If t is found during case 1 and 2, it can be showed the path of $\mathcal{R}$ can be extended by an L_1-shortest path to t. Hence, it guarantees the competitive ratio of $\sqrt{2}$ concerning the L_2-shortest-path.

Regarding case 3 (see Figure 3(b)), since I_p is in the middle quadrant q', $\mathcal{R}$ approaches to q'. If t is behind a gap in q', it will be similar to case 1, and the robot's path is an L_1-shortest-path. However, if t is behind one of the crucial gaps, $\mathcal{R}$ is getting away from t. So, the worst case is when the case 3 arises and t is behind a crucial gap; we denote the start point of such a case by p. After p, $\mathcal{R}$ may face many case 3 continuously because of some disappearance events. We consider all of them together and denote the endpoint of the last one by p'. We denote by p'' the point on the inflection ray of the reflex vertex r related to that crucial gap which is perpendicular to p. According to Figure 5(b), $\mathcal{R}$ travels $2x+x+y+z$ while the length of the shortest-path is $\sqrt{x^2+y^2}+z$. So, the competitive ratio is at most

$$\max\left\{\frac{2x+x+y+z}{\sqrt{x^2+y^2}+z}\right\}.$$

If we set $z=0$, by finding the function's extremum point, we get $\sqrt{10}$ as the maximum possible competitive ratio. Please note that if t is not visible at p', after p' $\mathcal{R}$ may face other cases until t is visible, but in those cases, the competitive ratio is less and greatest deviation occurs when t is found in a case 3. □

In the following theorem, we show that if the target point t is located in the same position of v, then the presented strategy guarantees an optimal competitive ratio.

Theorem 5 *If $t = v$, the competitive ratio is at most $\sqrt{2}$ and it is optimal.*

Proof. The proof is coming from Theorem 2 and Observation 1. In other words, in Observation 1 we showed that v always lies in I_p (for any $p \in path(s, t')$), and in Theorem 2 we showed that the strategy guarantees an L_1-shortest-path toward I_p. Since $t = v \in I_p$ Then it gives a competitive ratio of at most $\sqrt{2}$, which is optimal [6]. □

Efficient Algorithms for the k-colored Rainbow Sets

Hamidreza Keikha* Vahideh Keikha† Ali Mohades‡

Abstract

In this note, we introduce a variant of the minimum diameter color spanning set (MDCSS) problem. Let P be a set of n points of m colors in $\mathbb{R}^d$. The MDCSS problem is to find a subset of P that contains all the colors and admits the minimum possible diameter. Such a subset is called a rainbow set. Our objective is to find a rainbow set of size $k < m$ (so-called a k-rainbow set). This problem has application in weighted points, with the objective of finding a maximum weight k-rainbow set. First, we assume the points have similar weight and design an FPT algorithm, which we could generalize to the weighted version. We also solve the decision version and the reporting version of the problem by introducing a reduction to all maximal independent sets of a bipartite graph. We also provide almost 1.154-approximation algorithms for this problem. In $\mathbb{R}^2$, our exact algorithms have a complexity being near-linear to n.

1 Introduction

Let $P = \{p_1, \ldots, p_n\}$ be a set of n points in $\mathbb{R}^d$, the *diameter* of P is defined as $\max_{p_i,p_j \in P} d(p_i, p_j)$, and can be computed in $O(n \log n)$ time [12]. Now suppose each $p_i \in P$ is assigned a color. The objective of the *minimum diameter color spanning set (MDCSS)* problem is to find a subset P^* of P that contains one point from each color, and P^* has the smallest possible diameter among all choices of P^*, where the diameter is the maximum distance between any two points in P^*. See Figure 1 for an illustration. The set P^* is called the *color spanning* set or the *rainbow* set. The fixed-parameter tractability of MDCSS is posed as an open problem in [6], in which they assume that the dimension d is fixed. Very recently, in [11] this question is answered by proving that MDCSS is in W[1]-hard by using a complicated reduction from multi-colored clique graph problem [5], where the dimension d is not fixed. Also, the author shows that the problem does not admit an FPTAS in arbitrarily high dimensional spaces.

*Department of Computer Science, Sistan and Baluchestan University, Iran, keikha.eng@gmail.com

†The Czech Academy of Sciences, Institute of Computer Science, Czech Republic, keikha@cs.cas.cz

‡Department of mathematics and Computer Science, Amirkabir University, Iran, mohades@aut.ac.ir

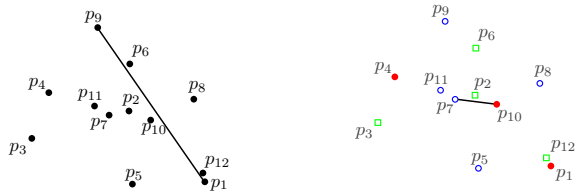


Figure 1: (left) The diameter of a set P of 12 points in $\mathbb{R}^2$. (right) For a set P with 3 colors, the rainbow set $P^* = \{p_2, p_7, p_{10}\}$.

In the same paper, some algorithms with quadratic dependencies to n are also supporting the result.

Minimum Color Spanning Circle (MCSC) For a set of n colored points of m colors, a closely related problem is the smallest color spanning circle, i.e. a circle of the smallest radius that is covering m distinct colors [1]. In $\mathbb{R}^2$, the smallest color spanning circle of m colors can be computed in $O(nm \log n)$ time by computing the upper envelope of some Voronoi surfaces [7, 1]. This problem becomes NP-hard in $\mathbb{R}^d$, where d is in the input, but admits a $(1+\epsilon)$-approximation in $O(dn^{\lceil 1/\epsilon \rceil + 1})$ time [8].

To the best of our knowledge the weighted version of these problems are not studied so far.

Our Contribution. We study a closely related variant of the MDCSS problem. In the following, we formally define our problems.

Definition 1 ***Minimum Diameter k-Colored Spanning Set (MDkCSS)*** *Let P be a set of n points of m colors. The objective of the MDkCSS problem is to find a subset $P^* \subset P$ of size $1 < k < m$ of distinct colors, such that P^* has the smallest possible diameter among all possible choices of P^*. We call P^* a k-rainbow set of P.*

See Figure 2. The main application of this problem is the case where the points have weight and the optimal k-rainbow set has the maximum weight. We define a *maximum weight k-rainbow set* P^* as a k-rainbow set that minimizes a parameter $\frac{d}{W}$, where d is the diameter and W is the total sum of the weighs of the points in P^*. These problems have also application in facility location, spatial database queries and the environments in which adjusting k goals of m goals suffices. In this note, we focus on the case where all the points have a similar weight. We achieve the following results:

- We introduce a relation between the MDCSS prob-

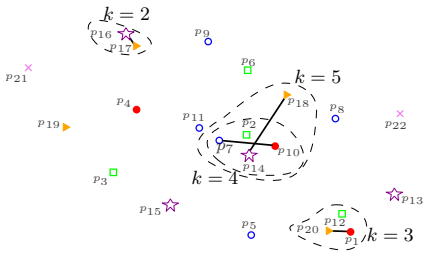


Figure 2: Problem definition and optimal solutions with k-rainbow sets for $k = 2, 3, 4, 5$.

lem and higher-order Voronoi diagrams. We design an FPT algorithm that has near-linear dependency on n in $\mathbb{R}^2$ (Theorem 2), which sounds to be helpful to improve the existing quadratic FPT algorithm for the MDCSS problem [11].

- We show that MDkCSS problem is fixed-parameter tractable in d-dimensional space for any fixed d (Theorem 3). This and previous result carry over to the weighted version of the problem.

- We study the decision version and the reporting version of MDkCSS problem for a given value $q > 0$ and introduce an $O(n(tk)^{5.5}\alpha)$ time algorithm, where α is the maximum number of the k-subsets of diameter at most q (Section 4).

- In $\mathbb{R}^2$, a 1.154-approximation of the k-rainbow set can be computed in $O(n^2mk^2 \log n)$ time. In $\mathbb{R}^d$, a $(1.154 + \epsilon)$-approximation can be computed in $O(dn^{\lceil 1/\epsilon \rceil + 2}k^d)$ time (Section 5).

We also remark that in MDCSS problem if the number of the existing colors in P is a small k (possibly constant), we still do not have any exact algorithm with a running time better than $\binom{n}{k}$. In $\mathbb{R}^2$, our FPT algorithm is near-linear to n.

2 Preliminaries

Let t denote the maximum frequency of any color in P. We first note that the proof in [6] shows NP-hardness for t bounded by three, and can easily be extended to also show NP-hardness if at most 5 colored points are co-located (if we do a reduction by MAX-E3SAT(5)). Consequently, the problem may get easier if the number of colors is large, i.e., more than $n/3$. See Appendix 7 for the details.

Proposition 1 *In MDCSS problem the maximum frequency of the colors does not appear in the hardness of the problem. Hence, the number of existing colors in P is a parameter that determines the hardness.*

Thereafter, one may assume $t \in O(1)$ and parameterize the complexity of the problem on other parameters, including the number of the colors in the rainbow set.

3 MDkCSS is in FPT in Any Fixed Dimension

Recall that a k-rainbow set $P^* \subset P$ is a set of points of k colors, where P^* has the smallest possible diameter among all color spanning subsets of size k. In [6] it is posed as an open question that which value of k is the threshold between easy and hard. We partially answer that question, as we not need to cover all, but only k colors that their instances realize the smallest possible diameter. Our algorithm has a near-linear dependency on the number of points, where its hardness depends on k (and t, but we discussed above that t is not a parameter to determine the hardness of the problem). Consequently, we answer the posed question in [6] partially as follows: for any constant number of colors which we need to cover in $\mathbb{R}^2$, the MDCSS problem can be answered in near-linear time, but for large values of k, the problem remains hard.

k-Order Voronoi Diagram. We recall that the *Voronoi diagram* of *order* k of P is the partitioning of the plane into a set of Voronoi cells, such that each Voronoi cell c is associated with a set $X \subseteq P$ of k points, and for each point p in the cell c, the k nearest neighbors of p are exactly the elements of X.

We denote this diagram by V_k. Such diagrams can be computed in $O(k^2n + n \log n)$ time and has at most $O(nk)$ cells [9].

In the following, we will show that any set of k colored points of smallest diameter is a subset of the points which are associated to a Voronoi cell of a Voronoi diagram of order $t(k-1)+1$, or $3t(k-1)+1$.

Lemma 1 *Let P be a set of n colored points with t as the maximum frequency of a color, and let P^* be a k-rainbow subset of P. Then P^* is a subset of the points which are corresponding to a Voronoi cell of a Voronoi diagram either of order $t(k-1)+1$, or $3t(k-1)+1$.*

Proof. Let $c(P)$ denote a subset of points of P that are associated with only one cell c of a Voronoi diagram $V_{t(k-1)+1}$, or $V_{3t(k-1)+1}$. Recall that for each Voronoi cell c, there exists a disk D having its center within c, where D contains no other point of $P - c(P)$. The set P^* also realizes a disk D^* such that either two or three points of P^* are located on its boundary.

Suppose by contradiction that the lemma is false, and P^* of k colors is not associated with one cell of a Voronoi diagram of order $t(k-1)+1$, or $3t(k-1)+1$.

By definition, in $V_{t(k-1)+1}$, the points of each cell of the diagram have the same $t(k-1)+1$ nearest neighbours in P. Observe that in the case where there are two points on the boundary of D^*, D^* cannot contain

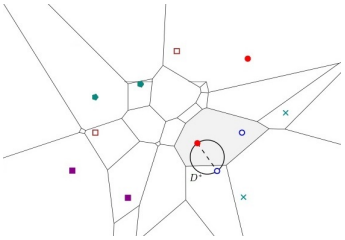


Figure 3: Illustration of Lemma 1, the case where D^* has two points on its boundary. On a set P of colored points with $t = 2$, the optimal solution with $k = 2$ is associated with a cell c (shown in gray) of $V_{t(k-1)+1}$, and uses a pair of red and blue points (connected by a dashed line segment). Observe that D^* cannot contain more than 3 points of P, otherwise, there must be two points of different colors strictly within D^*, such that they realize a smaller diameter than the diameter of D^*.

more than $t(k-1)+1$ points. If not, there always exist at least another set P' of k points from k distinct colors, which they all are entirely located within D^*, and the diameter of P' is strictly smaller than the diameter of D^* (i.e., P^*). This gives a contradiction. It follows that D^* cannot contain more than $t(k-1)+1$ points and P^* is contained in some Voronoi cell of a Voronoi diagram of order $t(k-1)+1$. See Figure 3 for an illustration.

In the case where D^* has three points of P^* on its boundary, we partition D^* into at most sectors by connecting the centre of D^* to the points of P^* on its boundary. Then obviously each sector cannot contain more than $t(k-1)$ points since otherwise there would be k points from distinct colors in that sector so that the determined diameter by those points is strictly smaller than the diameter of P^*. Then D^* is contained in some Voronoi cell of a Voronoi diagram of order $3t(k-1)+1$. □

3.1 Algorithm

From Lemma 1, the smallest diameter among each subset of k points of distinct colors that is associated to a Voronoi cell of $V_{t(k-1)+1}$ or $V_{3t(k-1)+1}$ determines an optimal solution. A similar idea is used in [2] to find a k-subset with the smallest diameter.

In the algorithm, we first make two Voronoi diagrams of orders $t(k-1)+1$ and $3t(k-1)+1$, of all the n points of P, without considering their colors in the construction. First consider the optimal solution on the Voronoi diagram of orders $3t(k-1)+1$. In each step of the algorithm, we consider the associated points of each cell of $V_{3t(k-1)+1}$ independently. Let P_c denote the associated points of a cell c. We use a brute force idea on P_c, to find a subset $P_c^* \subseteq P_c$ of k distinct colors with the smallest possible diameter, and remember the P_c^* with the smallest possible diameter among all the cells of $V_{3t(k-1)+1}$. We repeat the above algorithm also for $V_{t(k-1)+1}$, and report a set P_c^* with the smallest diameter.

In Lemma 1 we observed that each set P_c has a reasonable size with only a linear dependency to k and t, which makes our algorithm works exponentially only in k and t. Since the complexity of the number of the cells of a Voronoi diagram of order tk is $O(ntk)$, our method gives an FPT algorithm. In Appendix 6, we discuss the complexity of the algorithm. To generalize our FPT to a higher dimension d, we first need to construct the k-order voronoi diagram in that dimension. A k-order voronoi diagram in $\mathbb{R}^d$ can be constructed in $O(n^{\lceil d/2 \rceil} k^{\lfloor d/2 \rfloor + 1})$ time [3]. The other commands of the algorithm are the same as 2-D case.

Theorem 2 *Let P be a set of n colored points. The MDkCSS can be solved in $O(n2^{O(tk)} + n \log n)$ time, where k and t are the number of the colors in the rainbow set and the maximum frequency of a color, respectively.*

Proof. Computing a Voronoi diagram of order $O(tk)$ takes $O((tk)^2 n + n \log n)$ time and has at most $O(ntk)$ cells [9]. By repeating the algorithm of Section 3.1 for all the cells of the Voronoi diagrams V_{tk} and $V_{3t(k-1)+1}$, the algorithm takes $O(n2^{O(tk)} + n \log n)$ time. □

Theorem 3 *The MDkCSS problem is fixed-parameter tractable in d-dimensional space for any fixed d.*

Proof. A k-order voronoi diagram in the dimension d can be constructed in $O(n^{\lceil d/2 \rceil} k^{\lfloor d/2 \rfloor + 1})$ time [3]. If the dimension d of the MDkCSS problem is fixed, the problem can be solved in $O(n(2^{O(tk)} + \log n) + n^{\lceil d/2 \rceil} k^{\lfloor d/2 \rfloor + 1})$ time, which implies that MDkCSS is in FPT, where the frequency of the colors and the number of the colors are the fixed parameters of the problem. □

Maximum Weight k-rainbow Set Let w_i denote the weight of $p_i \in P$. A maximum weight k-rainbow set P^* can also be computed with the same algorithm since we consider all possible k-subsets. Instead of reporting the k-rainbow set of the smallest diameter d, we report the one which minimizes $\frac{d}{W}$, where $W = \sum_{p_i \in P^*} w_i$.

4 Reporting all MDkCSS of diameter at most q

In this section, we study the following problems: for a given P and $q > 0$, is there any k-rainbow set in P of diameter at most q, and how much quicker we can count or report all the k-rainbow sets of P of diameter at most q. Let c be any cell of $V_{3t(k-1)+1}$ that has at most $O(tk)$ points, and let $X \subseteq P$ denote the associated points of P to c. For any pair $p_i, p_j \in X$ of distinct colors, let $z = d(p_i, p_j)$ denote their Euclidean distance. Our objective is to determine whether is there any set of k points of distinct colors in X, where the pairwise

distances between the points are at most $z \leq q$. Consider two circles C_i and C_j of radius z, one is centered at p_i and the other at p_j. Let X' denote the set of points in $C_i \cap C_j \cap P$. Make a graph G on X' by connecting any pair of points with a distance at most z, and let $\overline{G}$ denote the complement of G. Observe that the vertices of X' which are lying on exactly one side of p_ip_j are at a distance less than z. Consequently, in $\overline{G}$, the connected pair of vertices lie on opposite sides of p_ip_j. This implies that we can consider $\overline{G}$ as a bipartite graph, with vertices at each side of p_ip_j as one part. Observe that the points of a distance smaller than z make a clique in G, and an independent set in $\overline{G}$. But now, the question is to check whether is there any maximal independent set (MIS) X^* of size at least k in $\overline{G}$, where at least k vertices in X^* has distinct colors. For each MSI, we check whether is there any set of k distinct colors among the reported vertices or not. By considering the freedom of p_i and p_j in $O(ntk)$ cells of $V_{3t(k-1)+1}$ (resp. $V_{t(k-1)+1}$), the algorithm takes $O(ntk) \times O(tk)^2 \times O((tk)^{2.5}\alpha)$ time, where α is the maximum number of the maximal subsets of diameter at most q.

Theorem 4 *Let P be a set of n colored points with t as the maximum frequency of a color. The reporting version of the MDkCSS can be answered in $O(n(tk)^{5.5}\alpha)$ time, where α is the maximum number of the maximal subsets of P with a diameter at most q.*

5 Approximation Algorithm

We design a simple but efficient approximation algorithm. Observe that for any set X of points, the diameter of X is longer than $\sqrt{3}$ times the radius of the smallest enclosing circle (SEC) of X; consider the configuration in which three points on the boundary of the SEC make an equilateral triangle, and the side of the triangle determines the diameter. If one translates any pair of these points on the boundary of the SEC, to get closer, the size of the diameter can only be increased. Let r_X and d_X denote the radius of the SEC and the diameter of X, respectively. For a set P of points, let X be the set realizing the smallest color spanning circle with k colors, and let P^* denote the set of points realizing the k-rainbow set of smallest diameter. Using the fact that the radius r_X is smaller than the radius of the color spanning circle of P^*, we have $d_{P^*} \leq d_X \leq 2r_{P^*} \leq 2/3\sqrt{3}(\sqrt{3}r_{P^*}) \leq 2/3\sqrt{3}d_{P^*}$.

Consequently, the diameter of the set X approximates the optimal k-rainbow set within a factor $2/3\sqrt{3} \approx 1.154$. A brute-force $O(n^4)$ time algorithm for computing the smallest color spanning circle of at least k colors consider any pair or triple of points of distinct colors that define a circle. But we can do better. According to Theorem 1.2. of [10], for a set of n points, computing a circle of smallest radius that satisfies m constraint (here covers m distinct colors) can be reformulated to satisfy only k of m constraints, in $O(nk^d)$ time, where d equals the geometric dimension of the original problem, and this would be performed by finding the optimal solution of $O(k^d)$ independent LP-type problems. Consequently, the presented algorithms for MCSC give approximation algorithms for our problems.

Theorem 5 *Let P be a set of n colored points of m colors. In $\mathbb{R}^2$, a 1.154-approximation of the k-rainbow set can be computed in $O(n^2mk^2 \log n)$ time. In $\mathbb{R}^d$, a $(1.154+\epsilon)$-approximation can be computed in $O(dn^{\lceil 1/\epsilon \rceil + 2}k^d)$ time.*

Acknowledgement V.Keikha is supported by the Czech Science Foundation, grant number GJ19-06792Y, and with institutional support RVO:67985807.

References

[1] M. Abellanas, F. Hurtado, C. Icking, R. Klein, E. Langetepe, L. Ma, B. Palop, and V. Sacristán. Smallest color-spanning objects. In *European Symposium on Algorithms*, pages 278–289. Springer, 2001.

[2] A. Aggarwal, H. Imai, N. Katoh, and S. Suri. Finding k points with minimum diameter and related problems. *Journal of Algorithms*, 12(1):38–56, 1991.

[3] K. L. Clarkson and P. W. Shor. Applications of random sampling in computational geometry, ii. *Discrete & Computational Geometry*, 4(5):387–421, 1989.

[4] U. Feige. A threshold of ln n for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.

[5] M. R. Fellows, D. Hermelin, F. A. Rosamond, and S. Vialette. On the parameterized complexity of multiple-interval graph problems. *Theoretical computer science*, 410(1):53–61, 2009.

[6] R. Fleischer and X. Xu. Computing minimum diameter color-spanning sets is hard. *Information Processing Letters*, 111(21):1054 – 1056, 2011.

[7] D. P. Huttenlocher, K. Kedem, and M. Sharir. The upper envelope of voronoi surfaces and its applications. *Discrete & Computational Geometry*, 9(3):267–291, 1993.

[8] M. R. Kazemi, A. Mohades, and P. Khanteimouri. On approximability of minimum color-spanning ball in high dimensions. *Discrete Applied Mathematics*, 279:188–191, 2020.

[9] D.-T. Lee. On k-nearest neighbor voronoi diagrams in the plane. *IEEE transactions on computers*, 100(6):478–487, 1982.

[10] J. Matoušek. On geometric optimization with few violated constraints. *Discrete & Computational Geometry*, 14(4):365–384, 1995.

[11] J. Pruente. Minimum diameter color-spanning sets revisited. *Discrete Optimization*, 34:100550, 2019.

[12] G. T. Toussaint. Solving geometric problems with the rotating calipers. In *Proc. IEEE Melecon*, volume 83, page A10, 1983.

Appendix

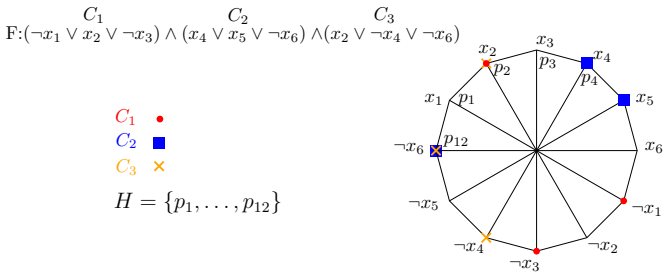


Figure 4: Construction of an instance X of MDCSS problem from a satisfying problem F, by a regular polygon H. Note that in F, each variable must exactly occur in 5 clauses, which here we omit this assumption for the clarity. In a complete construction, we will have exactly 5 points of different colors at the vertices of H that contribute to X.

6 Running Time Analysis

In the following, we consider the complexity of running the algorithm on each cell of $V_{3t(k-1)+1}$. The analyze of running the algorithm on each cell of $V_{t(k-1)+1}$ is similar. To analyze the running time, we use Stirling's formula as below

$$\binom{3tk-3t+1}{k} = 2^{\log(3tk-3t+1)!-\log k!-\log(3tk-3t-k+1)!}$$

$$\log(3tk-3t+1)! - \log k! - \log(3tk-3t-k+1)! =$$
$$3tk\ln(3tk-3t+1) - (3tk-3t+1) + O(\ln(3tk-3t+1))$$
$$-k\ln k + k - O(\ln k) - (3tk-3t+1)\ln(3tk-3t-k+1)$$
$$+(3tk-3t-k+1) - O(\ln(3tk-3t-k+1)) \in O(tk)$$

Consequently, considering all possible k-rainbow sets corresponding to one cell of $V_{3t(k-1)+1}$ takes $O(2^{tk})$ time.

For each cell c of a Voronoi diagram of order $V_{3t(k-1)+1}$, we can find a k-rainbow set with the smallest possible diameter among the corresponding points of c in $O(k{\cdot}k\log k\cdot 2^{tk})$ time, in which, in $O(k)$ time we determine whether the selected set contains k distinct colors, and $O(k\log k)$ time is required to find the diameter of a set of size k. Note that the first and the second terms of the running time will be dominated by the last term.

7 Discussion on the hardness of MDCSS

Theorem 6 *MDCSS remains NP-Complete even if at most 5 colored points are co-located.*

Proof. The proof is the same as the reduction in [6]. Here we recall it by doing the reduction on a regular polygon to avoid the careful placement of the points on the circle as is required in [6]. We use a reduction from MAX-E3SAT(5) that is a variant of MAX-3SAT, in which each clause contains exactly 3 literals, every variable occurs in exactly 5 clauses and a variable does not appear in a clause more than once, which is known to be NP-hard [4].

For a MAX-E3SAT(5) instance F of n variables and m clauses, we make an instance of MDCSS with a set X of m colors, in which the diameter of the rainbow set of X will be strictly smaller than 1 if F is satisfiable. We construct the set X from the variables of F as below.

Consider a regular polygon $H = \{p_1, \ldots, p_{2n}\}$ of diameter 1 and with $2n$ vertices, where the antipodal points of each vertex p_i, which is p_{n+i} also exists on H. We associate each p_i and p_{n+i} to exactly one literal of F.

Each vertex and its antipodal vertex determine a positive literal and the negative literal which is assigned to this pair of antipodal vertices. We assign all the literals of F on the vertices of H. We also assign a unique color r_j to each clause C_j. For each variable $x_i \in F$, if x_i appears in C_j, we place a point of color r_j at vertex p_i, and if $\neg x_i$ appears in C_j, we place a point of color r_j at vertex p_{n+i}. Then X is the set of all the colored points which we have placed on the vertices of H. See Figure 4 for an illustration.

If F is satisfiable, then we can find a correct assignment for all the variables, such that for none of them, both of the true and the false value appears in the assignment. Consequently, this assignment has a diameter strictly smaller than 1, since the diameter of H has the maximum possible value on the antipodal pairs. Also, this assignment has satisfied all the clauses, and consequently, introduces a color spanning set, and also answers the decision version of the existence of a MDCSS with a diameter strictly smaller than 1.

If X contains a rainbow set X^* of m points of a diameter strictly smaller than 1, then X^* cannot contain both p_i and p_{n+i}. We can use the set X^* to find a correct satisfying assignment for the literals of F. Consequently, F is satisfiable if and only if a rainbow set X^* of m points of diameter smaller than 1 exists. □

Remark. The reduction can be done in $O(n+m)$ time. When necessary, one can easily justify the assumption of having exactly t copies of a color by considering dummy points in the infinity distance.

Proof of Theorem 5

Proof. We use the algorithm of [7, 1] for the first result, and the algorithm of [8] for the second result. □

A lower bound on the stretch factor of Yao graph Y_4

Davood Bakhshesh* Mohammad Farshi†

Abstract

In this paper, we provide a lower bound of 3.828 for the stretch factor of Y_4. This solves an open problem posed by Barba et al (*L. Barba et al., New and improved spanning ratios for Yao graphs. Journal of computational geometry*, 6(2):19–53, 2015.).

1 Introduction

Let S be a set of points in the plane. A weighted graph $G = (S, E)$ is called *geometric*, if every edge $e = (p, q)$ in G is a straight-line between p and q and the weight of e is $|pq|$, the Euclidean distance between p and q. The length of a path between two points $p, q \in S$ is defined as the sum of the weight of all edges of the path. For any two points $p, q \in S$, the *stretch factor* of the pair p and q is the ratio of the length of a shortest path between p and q over $|pq|$. The *stretch factor* (or *dilation*) of G is the maximum stretch factor between any pair of vertices of G. For a real number $t > 1$, a geometric graph G is called a *t-spanner* if the stretch factor of G is at most t.

One of the most popular graphs in computational geometry are *Yao graphs*, denoted by Y_k, introduced by Flinchbaugh and Jones [10] and independently by Yao [12]. For every point set S in the plane and an integer $k \geq 2$, the Yao graph Y_k is constructed as follows. Around each point $p \in S$, the plane is partitioned into k regular cones with apex at p. We denote the set of all these cones by $\mathcal{C}_p$. Then, for each cone $C \in \mathcal{C}_p$, we add an edge (p, r) to Y_k, where r is a closest point in C to p. Throughout the paper, we call such a point r by the **nearest** point to p in C_p. Another popular graphs are *theta-graphs* (Θ_k) which were introduced by Clarkson [6] and independently by Keil [11]. The construction of Θ_k is similar to the construction of Yao graph Y_k except that we change the definition of **nearest** as follows: A point $r \in C_p$ is the **nearest** point to p if the orthogonal projection of r, denoted by r', onto the bisector of C_p minimizes $|pr'|$. For any two vertices p and q, we denote by $q \in C_p$, if the cone with the angular diameter θ and apex at p contains q, and the *canonical triangle* T_{pq} is defined an isosceles triangle such that the angle of its apex at p is θ and its base is perpendicular to the bisector of C_p.

Clarkson [6] proved that Y_{12} is a $(1 + \sqrt{3})$-spanner. Then, Althöfer at al. [1] proved that there is an integer k such that Y_k is a t-spanner for every $t > 1$. For $k > 8$, Bose et al. [5] showed that Y_k is a t-spanner with $t \leq 1/(\cos\theta - \sin\theta)$, where $\theta = \frac{2\pi}{k}$ Then, Bose et al. [4] showed that for $k > 6$, Y_k is a $1/(1 - 2\sin(\theta/2))$-spanner. Damian and Raudonis [8] proved Y_6 has the stretch factor at most 17.64. This was later improved by Barba et al. to 5.8 [2]. The authors of [2] also improved the stretch factor of Y_k for all odd values of $k \geq 5$ to $1/(1 - 2\sin(3\theta/8))$. Bose et al. [4] proved that Y_4 is a 663-spanner. Damian et al. [7] improved the upper bound of 663 on Y_4 to 54.62. In [2], Barba et al. provided the lower bounds of 2.87 and 2 on the stretch factor of Y_5 and Y_6, respectively. We summarized some the previous results on the stretch factor of Yao graphs in Table 1.

Y_k	Lower bound	Upper bound
Y_6	2 [2]	5.8 [2]
Y_5	2.87 [2]	3.74 [2]
Y_4	**3.828 (this paper)**	54.62 [7]
Y_2, Y_3	∞	Not a spanner [9]

Table 1: Lower and upper bounds on the stretch factor of Y_k.

In this paper, we provide a lower bound of 3.828 on the stretch factor of Y_4. This solves an open problem posed by Barba et al. [2].

2 Main Result

Here, we provide a lower bound of 3.828 for the stretch factor of Y_4. We present a point set S that the graph Y_4 on S has the stretch factor 3.828. The construction of S is a modification of the construction of the point set S' which was proposed by Barba et al. [3] for Θ_4. Let $C_0(a), C_1(a), C_2(a)$ and $C_3(a)$ be four regular cones with apex at a of angle 90° that equally partition the plane. We assume that the first ray of $C_0(a)$ is in the direction of positive x-axis. We assume that the cones $C_0(a), C_1(a), C_2(a)$ and $C_3(a)$ are in counterclockwise order around a. Let $D_i(a, b)$ be a quarter of a closed disk with center a and radius $|ab|$ such that cone $C_i(a)$ contains $D_i(a, b)$ (see Figure 1(a)). The con-

*Department of Computer Science, University of Bojnord, Bojnord, Iran. d.bakhshesh@ub.ac.ir

†Combinatorial and Geometric Algorithms Lab., Department of Mathematical Sciences, Yazd University, Yazd, Iran. mfarshi@yazd.ac.ir

struction is started with $S = \{u, w\}$, where u and w are two points such that $C_1(u)$ contains w and the difference of x-coordinates of u and w is arbitrarily small. Now, it is clear that the graph Y_4 on S has a single edge (u, w). Then, the stretch factor of Y_4 on S is 1. The idea to obtain the lower bound 3.828 is that we extend the shortest path between u and w by adding some extra points to S. Now, let x be a point in the plane such that x is inside $D_1(u, w)$ and lies at the corner of $D_1(u, w)$ (see Figure 1(b)). Now, we add x to S. Let r be a point in the plane such that r is inside $D_3(w, u)$ and lies at the corner of $D_3(w, u)$ as depicted in Figure 1(b). Then, we add r to S. Now, we have $S = \{u, w, x, r\}$. If we draw the graph Y_4 on S, then we have not the edge (u, w) and therefore the shortest path between u and w has been extended. Now, by the above technique, we extend the shortest path between u and w as depicted in Figure 1(b). Note that y is a point placed at a corner of $D_0(x, w)$ with $|xy| < |xw|$, and z is a point placed at a corner of $D_3(y, w)$ with $|yz| < |yw|$ (see Figure 1(b)). Notably, we supposed that $|uw| = 1$. Then, by the construction, we have

$$|ux| \approx |uw| = 1$$
$$|xy| \approx |xw| \approx \sqrt{2}$$
$$|yz| \approx |ux| \approx 1$$
$$|yw| \approx 2\sqrt{2}\sin\frac{\pi}{8} \quad (yw \text{ is the base side of } \triangle_{xyw})$$
$$|zw| \approx \sqrt{|yw|^2 - |ux|^2} = \sqrt{\left(2\sqrt{2}\sin\frac{\pi}{8}\right)^2 - 1} = \sqrt{2} - 1.$$

Then,

$$\frac{|ux| + |xy| + |yz| + |zw|}{|uw|}$$
$$\approx \frac{1 + \sqrt{2} + 1 + \sqrt{2} - 1}{1} = 2\sqrt{2} - 1 \approx 3.828$$

Hence, we proved the following theorem.

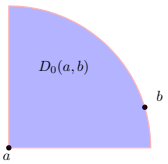
Theorem 1 *The stretch factor of Y_4 is at least 3.828.*

3 Conclusion

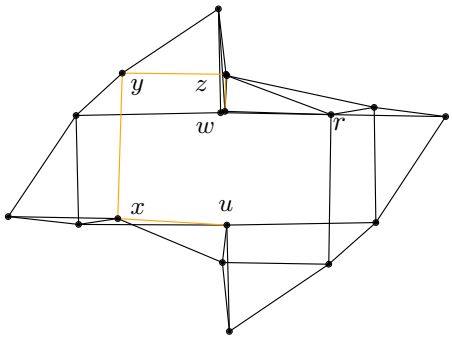
In this paper, we provided a lower bound of 3.828 for the stretch factor of Y_4 that solves an open problem. It is an interesting open problem to know if 3.828 is the best lower bound for the stretch factor of Y_4.

References

[1] I. Althöfer, G. Das, D. Dobkin, D. Joseph, and J. Soares. On sparse spanners of weighted graphs. *Discrete & Computational Geometry*, 9(1):81–100, 1993.



(a) $D_0(a, b)$.



(b) A lower bound of Y_4. The orange path is the shortest path between u and w.

Figure 1: $D_0(a, b)$ and the graph Y_4.

[2] L. Barba, P. Bose, M. Damian, R. Fagerberg, W. L. Keng, J. O'Rourke, A. van Renssen, P. Taslakian, S. Verdonschot, and G. Xia. New and improved spanning ratios for Yao graphs. *Journal of computational geometry*, 6(2):19–53, 2015.

[3] L. Barba, P. Bose, J.-L. De Carufel, A. van Renssen, and S. Verdonschot. On the stretch factor of the Theta-4 graph. In F. Dehne, R. Solis-Oba, and J.-R. Sack, editors, *Algorithms and Data Structures*, pages 109–120, Berlin, Heidelberg, 2013.

[4] P. Bose, M. Damian, K. Doueb, J. O'rourke, B. Seamone, M. Smid, and S. Wuhrer. $\pi/2$-angle Yao graphs are spanners. *International Journal of Computational Geometry & Applications*, 22(01):61–82, 2012.

[5] P. Bose, A. Maheshwari, G. Narasimhan, M. Smid, and N. Zeh. Approximating geometric bottleneck shortest paths. *Computational Geometry*, 29(3):233 – 249, 2004.

[6] K. Clarkson. Approximation algorithms for shortest path motion planning. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, STOC '87, pages 56–65, New York, NY, USA, 1987. ACM.

[7] M. Damian and N. Nelavalli. Improved bounds on the stretch factor of Y_4. *Computational Geometry: Theory and Applications*, 62:14 – 24, 2017.

[8] M. Damian and K. Raudonis. Yao graphs span Theta graphs. *Discrete Mathematics, Algorithms and Applications*, 04(02):1250024, 2012.

[9] N. M. El Molla. *Yao spanners for wireless ad hoc networks*. Master's thesis, Villanova University, 2009.

[10] B. Flinchbaugh and L. Jones. Strong connectivity in directional nearest-neighbor graphs. *SIAM Journal on Algebraic Discrete Methods*, 2(4):461–463, 1981.

[11] J. M. Keil. Approximating the complete Euclidean graph. In R. Karlsson and A. Lingas, editors, *SWAT 88*, pages 208–213, Berlin, Heidelberg, 1988.

[12] A. C.-C. Yao. On constructing minimum spanning trees in k-dimensional spaces and related problems. *SIAM Journal on Computing*, 11(4):721–736, 1982.

Local geometric spanners for points in convex position

Mohammad Ali Abam* Mohammad Sadegh Borouny†

Abstract

Let P be a set of n points in the plane that are in convex position. For a geometric graph G on a point set P and a region R belonging to a family $\mathcal{R}$, $G \cap R$ is defined to be the part of the graph G that is inside R. A local t-spanner with respect to $\mathcal{R}$ is a geometric graph G on P such that for any region R in $\mathcal{R}$, the graph $G \cap R$ is a t-spanner w.r.t. $G_c(P) \cap R$, where $G_c(P)$ is the complete geometric graph on P. We prove that the set P for any constant $\varepsilon > 0$ admits a local $(1+\varepsilon)$-spanner of size $O(n)$ and a local $(2+\varepsilon)$-spanner of size $O(n \log^2 n)$ w.r.t. vertical slabs and squares, respectively. Moreover, if adding Steiner points is allowed, a local $(1+\varepsilon)$-spanners with $O(n \log^2 n)$ edges can be obtained for rectangle regions. Also, if regions are arbitrary oriented squares, we can obtain a local $(1+\varepsilon)$-spanner of size $O(n \log^2 n)$ using $O(n \log n)$ Steiner points.

1 Introduction

Let $G(P, E)$ be an edge weighted geometric graph on a set P of n points in plane. The length of a shortest path and Euclidean distance of any node p and q are denoted by $d_G(p, q)$ and $|pq|$ respectively. G is a t-spanner (for the given $t > 1$) if for any two vertices $p, q \in P$ we have $d_G(p, q) \leq t \cdot |pq|$. Also G is called local t-spanner with respect to a family $\mathcal{R}$ of regions if for any region $R \in \mathcal{R}$ and any two points $p, q \in P \cap R$, the distance between p and q in $G \cap R$ is at most $t \cdot |pq|$.

In this paper, we consider spanners for point sets that are in a convex position. We focus on some specific families, namely vertical slabs, squares, rectangles, and present local t-spanners with near-linear sizes.

Even if the points are in convex position, G must have $O(n^2)$ edges to be local with respect to some families of regions such as slabs (see [2]). Hence, we shall consider the case where we are allowed to add Steiner points to the graph. In other words, instead of constructing a geometric network for P, we are allowed to construct a network for $P \cup Q$ for some set Q of Steiner points. We say that a graph G on $P \cup Q$ is a local Steiner t-spanner w.r.t. $\mathcal{R}$ for P, if for any $R \in \mathcal{R}$ and any two points $p, q \in P \cap R$, the distance between p and q in $G \cap R$ is at most $t \cdot |pq|$.

Abam *et al.* in [2] prove that any point set of n points in the plane for any constant $\varepsilon > 0$ admits a local $(4+\varepsilon)$-spanner of size $O(n \log^2 n)$ and a local $(1+\varepsilon)$-spanner of size $\Omega(n \log n)$ w.r.t. axis-parallel squares and vertical slabs, respectively. By adding Steiner points, they obtained a local $(1+\varepsilon)$-spanner w.r.t. disks of size $O(n \log^2 n)$ using $O(n)$ Steiner points.

In section 2, we construct a local $(1+\varepsilon)$-spanner of size $O(n)$ when regions are vertical slabs. In section 3, we consider square regions, and show it is possible to construct a local $(2+\varepsilon)$-spanner of size $(n \log^2 n)$ w.r.t. axis-parallel squares. Also, we consider arbitrary oriented squares and we show if Steiner points are allowed, we can construct a local $(1+\varepsilon)$-spanner of size $O(n \log^2 n)$. We dedicate Section 4 to rectangle regions, and show by adding $O(n \log n)$ Steiner points, we can construct a local $(1+\varepsilon)$-spanner of size $O(n \log^2 n)$.

2 Local spanners w.r.t. vertical slabs

For arbitrary slabs (not necessarily vertical) and any set of points in general position, we need $\Omega(n^2)$ edges to construct a local spanner because any edge of the complete graph can introduce a narrow slab which contains only its two endpoints [2]. Hence, we only focus on vertical slabs (i.e. $[a, b] \times [-\infty, \infty]$ for some a and b). Our method to construct local $(1+\varepsilon)$-spanners w.r.t. vertical slab is based on a well-separated-pair decomposition of P defined below.

Definition 1 *[4] Let P be a set of n points in the plane and let $s > 0$ be a real number. Two point sets A and B in the plane are well-separated w.r.t. s, if there are two disjoint disks D_A and D_B of the same radius, r, such that (I) D_A and D_B covers A and B, respectively (II) the distance between D_A and D_B is at least $s \cdot r$. A well-separated pair decomposition (WSPD) for P w.r.t. s (called s-WSPD) is a collection $\Psi := \{(A_1, B_1), \ldots, (A_m, B_m)\}$ of pairs of non-empty subsets of P such that*

1. *A_i and B_i are well-separated w.r.t. s, for all $i = 1, \cdots, m$.*
2. *for any two distinct points p and q of P, there is exactly one pair $(A_i, B_i) \in \Psi$, such that $p \in A_i$ and $q \in B_i$, or vice versa.*

*Department of Computer Engineering, Sharif University of Technology, Tehran, Iran abam@sharif.edu

†Department of Computer Engineering, Sharif University of Technology, Tehran, Iran borouny@ce.sharif.edu

Callahan and Kosaraju show that any set P admits a WSPD of size $m = O(s^2 n)$, where the number of pairs is called the size of the WSPD.

We will show that any set P in convex position admits a WSPD of size $m = O(s^2 n)$ such that for each pair (A, B) of the WSPD both A and B are continuous pieces of the lower hull or upper hull of P.

Our method is based on the WSPD construction by Fisher and Har-Peled [5]. Their construction uses a compressed quadtree, which is defined as follows. Let U be the bounding square for P. Partition U into four equal-sized squares. Continue recursively until each square in the final subdivision contains a single point. In our construction, we first rotate the axes such that the x-axis is parallel to pq where p and q are the leftmost and rightmost points on the x-axis. This causes that the lower and upper hulls get separated at the root of the quadtree. Moreover, each cell of the quadtree may contain a constant number of continuous pieces of the lower or upper hulls of P, so we divide each cell into a constant number of new cells; each containing one continuous piece of P.

It is known by setting $s := 4 + (8/\varepsilon)$, and selecting an edge per pair (A, B), we can construct $(1+\varepsilon)$-spanner. Unfortunately, this construction might not be local spanner. Indeed, for a region $R \in \mathcal{R}$, $R \cap A$ and $R \cap B$ might not be empty but the selected edge for pair (A, B) might not exist in R. To get a local spanner, for each pair (p, q) where $p \in A$ and $q \in B$, we add pq to the spanner provided that there is a region $R \in \mathcal{R}$ just containing p, q among points in $A \cup B$.

Now, after constructing our WSPD for point set P in convex position, for each pair (p, q) where $p \in A$ and $q \in B$, we add pq to the spanner G provided that there is a vertical slab just containing p and q among points in $A \cup B$. We will show that the number of edges of the constructed graph is $O(n)$.

Let (A, B) be a pair of the WSPD. If A and B can be separated by a vertical line, it is easy to see we only add one edge to G; meaning one edge per such a pair. Otherwise, we add an edge for any p and q such that $p \in A$ and $q \in B$ and p and q are consecutive on the x-axis among all points in $A \cup B$. Since A and B contain continuous pieces of the convex hull of P, so p and q must be consecutive on the x-axis among all points in P. This happens only $O(n)$ times. Therefore, the total edges added to G is $O(n)$. For each point pair as (p, q) there is a pair of WSPD as (A_i, B_i) such that $p \in A_i$ and $q \in B_i$ or vice versa. So we get the following theorem.

Theorem 1 *For a set P of n points in convex position and any $\varepsilon > 0$, there exists a local $(1+\varepsilon)$-spanner of size $O(n)$ w.r.t. vertical slabs.*

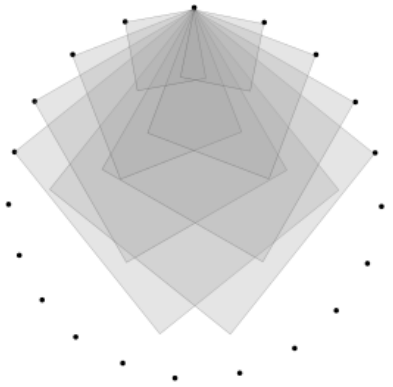


Figure 1: A configuration of points in convex position for which any local $(1+\varepsilon)$-spanner w.r.t. arbitrary oriented squares needs $\Omega(n^2)$ edges [2].

3 Local spanners w.r.t. squares

For arbitrary squares (not necessarily axis-parallel), there is an example showing that we may need $\Omega(n^2)$ edges to have a local spanner—see Fig. 1 [2]. Hence, in the first part, we only focus on axis-parallel squares, and in the second part, we consider the case where we are allowed to add Steiner points to construct a local spanner w.r.t. arbitrary oriented squares.

3.1 Local spanners w.r.t. axis-parallel squares

Our local spanner construction is based on the concept of Cone Separated Pair Decomposition [1], CSPD, defined next.

Let α-cone be a cone with angle α. Let C be a collection interior-disjoint α-cone whose apices are the origin and also they together cover the whole plane.

Let P be a set of points in the plane. A pair (A, B) $(A, B \subset P)$ is said to be cone separated if there exists a translated cone $\sigma \in C$ such that (i) all points of A are inside σ (ii) all points of B are inside the reflection of σ about its apex.

A CSPD of P with respect to α is a collection $\psi_\alpha := \{(A_1, B_1), \cdots, (A_m, B_m)\}$ of pairs of subsets for P such that:

- A_i and B_i are cone separated, for all $i = 1, \cdots, m$.
- For any two distinct points $p, q \in P$, there exists precisely one pair in $(A_i, B_i) \in \psi_\alpha$ such that (i) $p \in A_i$ and $q \in B_i$ or (ii) $q \in A_i$ and $p \in B_i$.

In [1] it has been shown that any set P of n points and any angle α admit a CSPD $\psi_\alpha = \{(A_1, B_1), \cdots, (A_m, B_m)\}$ such that $\sum |A_i| + |B_i| = O(n \log^2 n)$.

First, we simply divide P into four sets P_1, P_2, P_3 and P_4 such that each P_i is xy-monotone. It is easy to see

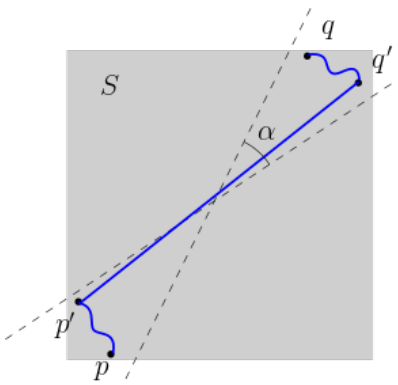


Figure 2: $p, p' \in P_1$ and $q, q' \in P_3$; also, $p, p' \in A$ and $q, q' \in B$ where (A, B) is a pair of α-CSPD. Blue path show $(2+\varepsilon)$ path.

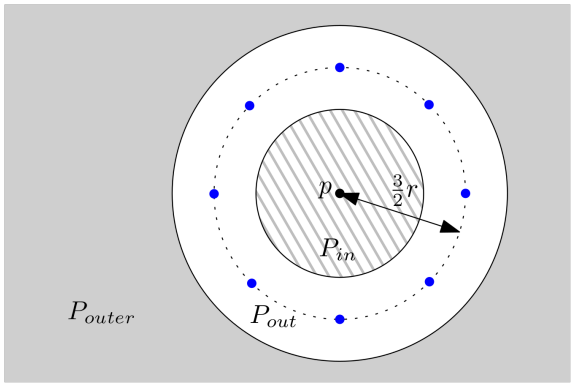


Figure 3: Steiner points between P_{in} and P_{outer}. Blue points show Steiner points.

that each square intersects a continuous piece of the convex hull of P_i. We can construct a local $(1+\varepsilon)$-spanner w.r.t. axis-parallel squares for each P_i of size $O(|P_i|)$ using the WSPD method as P_i is xy-monotone. Then, for each pair (P_i, P_j), we construct an α-CSPD where α depends on the given ε. Again we can construct this α-CSPD such that for each pair (A, B), both A and B are continuous pieces of the convex hull of P. For each pair (A, B) of the α-CSPD, we add the Delaunay edges $A \cup B$ w.r.t. squares. Note that the number of edges added to the spanner is $\sum O(|A| + |B|) = O(n \log^2 n)$.

We next show that for any two points p and q there is a $(2+\varepsilon)$-path inside any square S containing p and q. If p and q belong to set P_i for some i, we can easily show that there is a $(1+\varepsilon)$-path inside S. Now, assume $p \in P_i$ and $q \in P_j$ for some $i \neq j$. We know there is a pair (A, B) of the α-CSPD of $P_i \cup P_j$ such that $p \in A$ and $q \in B$. There should be points $p' \in A \cap S$ and $q' \in B \cap S$ such that there is an edge connecting p' to q' in the spanner. We know that there are $(1+\varepsilon)$-paths inside S from p to p' and from q to q'—see Fig 2. For appropriate α (depending on ε) we can show that these two paths together with edge $p'q'$ is a $(2+\varepsilon)$-path.

Theorem 2 *For any set P of n points in the convex position and any $\varepsilon > 1$, one can construct a local $(2+\varepsilon)$-spanner w.r.t. axis-parallel squares of size $O(n \log^2 n)$.*

3.2 Local spanners w.r.t. arbitrary oriented squares

Local spanners w.r.t. arbitrary squares are more challenging. Our method uses some definitions defined below.

Definition 2 *Let $b(p, r)$ be the set of all points of P whose distance to $p \in P$ is at most r. Also, let ring(p, r, R) is the set of all points of P whose distance to $p \in P$ is at most R and at least r.*

Lemma 3 *([3]) Let P be a set of n points in $\mathbb{R}^d$, $t > 0$ be a parameter, and let c be a sufficiently large constant. Then, one can compute in linear time a ball $b = b(p, r)$, such that (i) $|b \cap P| \geq n/c$, (ii) ring$(p, r, r(1+1/n))$ is empty, and (iii) $|P \backslash b(p, 2r)| \geq n/2$.*

Based on Lemma 3, let $P_{\text{in}} = b(p, r)$, $P_{\text{out}} = b(p, 2r) \backslash b(p, r)$ and $P_{\text{outer}} = P \backslash b(p, 2r)$. We define an angle $\theta = 2\pi/k$ depending on the given ε—note that k is constant depending on ε. We select k points on the circle centered at p and radius $\frac{3}{2}r$ such that for any two consecutive such points q and q', the angle qpq' is θ—see Fig 3. We look at all these points as Steiner points. Then, we connect each Steiner point to all other original points in our spanner; this in total adds $O(n)$ edges to the spanner. These edges guarantee that there is always a $(1+\varepsilon)$-path from a point in P_{in} to a point in P_{outer} even we cut the spanner by a square.

In the first step of our construction, we connect each Steiner point to all other points in our spanner. Since the number of Steiner points is constant, then the number of edges added to the spanner is $O(n)$. Moreover, it is easy to show that there is always a $(1+\varepsilon)$-path from a point in P_{in} to a point in P_{outer} even we cut the spanner by an oriented square. Next we explain how to guarantee the existing a $(1+\varepsilon)$-path from a point in P_{in} to a point in P_{out} by adding $O(n \log n)$ edges. If we succeed in doing that, then we can recursively continue our construction over P_{in} and $P_{\text{out}} \cup P_{\text{outer}}$ and get a $O(n \log^2 n)$ spanner.

To obtain a spanner over $(P_{\text{in}}, P_{\text{out}})$ we add new Steiner points. First, we construct an s-WSPD ($s \geq 2$ depends on ε) over $(P_{\text{in}}, P_{\text{out}})$. For each pair (A, B) of the WSPD, we add a constant number of Steiner points. Let o_A and r be the center and radius of the bounding disk of A. As the last step, we select k points on the circle centered at o_A with radius $\frac{3}{2}r$, and consider all these points as Steiner points add edges between them and all points in $A \cup B$ in our spanner—see Fig 4. Since

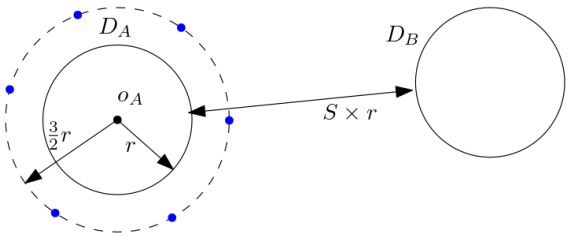


Figure 4: Steiner points for each pair (A, B) of the WSPD. Blue points show Steiner points.

$s \geq 2$, it is clear that a square containing non-empty A and B subsets must contain at least one Steiner point defined above. Since the number of pairs is $O(n)$ and we add $O(1)$ Steiner points per pair, the total number of Steiner points is $O(n)$. We show that the number of edges that are added to the spanner is $O(n \log n)$.

We use the fact that in the WSPD construction method, which we presented in section 2, if $\text{ring}(p, r, r(1+1/n))$ is empty, the depth of the quadtree for $P_{\text{in}} \cup P_{\text{out}}$ should become $O(\log n)$—note that we just want to separate P_{in} from P_{out} by WSPD. As the depth is $O(\log n)$, the sum $O(|A| + |B|)$ over all pairs is $O(n \log n)$, and therefore, the total edges added to the spanner is $O(n \log n)$. Putting all this together, we get the following theorem.

Theorem 4 *Suppose ε is a parameter and P is an arbitrary point set in convex position. One can construct a local $(1+\varepsilon)$-spanner with respect to square that uses $O(n \log n)$ Steiner point and has $O(n \log^2 n)$ edges.*

4 Local spanners w.r.t. rectangles

For axis-parallel rectangles, there is an example showing that we may need $\Omega(n^2)$ edges to have a local spanner—see Fig. 5. Hence, we only consider the case where we are allowed to add Steiner points to the spanner.

Our local spanner construction again is based on CSPD. We construct an α-CSPD for α depending on ε. For each pair (A, B) of the CSPD, let σ be an α-cone that together with its reflex about its apex, separates A and B. We consider the apex o of σ as a Steiner point and connect o to all points in $A \cup B$ in the spanner, giving us a spanner of size $O(n \log^2 n)$ edges. For any two points p and q, it is easy to see the path p, o, q is a $(1+\varepsilon)$-path for appropriate α. As o is inside any square containing p and q, then our spanner is a $(1+\varepsilon)$-local spanner.

Theorem 5 *For a set P of n points in convex position and any $\varepsilon > 0$, there exists a local $(1+\varepsilon)$-spanner of size $O(n \log^2 n)$ w.r.t. axis-parallel rectangles by adding at most $O(n \log^2 n)$ Steiner points.*

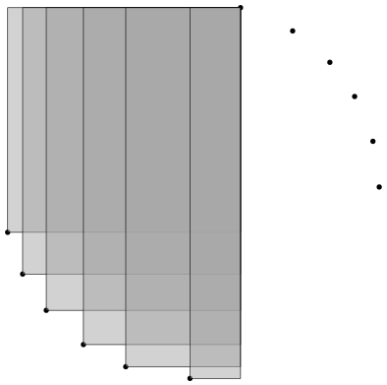


Figure 5: A configuration of points in convex position for which any local $(1+\varepsilon)$-spanner w.r.t. axis-parallel rectangles needs $\Omega(n^2)$ edges.

References

[1] M. A. Abam and M. de Berg Kinetic Spanners in $\mathbb{R}^d$. *Discrete Computational Geometry*, 45(4):723–736, 2011.

[2] M. A. Abam, M. S. Borouny and A. Mousavi Local Geometric Spanner. *ICCG'02*, 45–48, 2019.

[3] M. A. Abam and S. Har-Peled New constructions of SSPDs and their applications. *Comput. Geom*, 45(5-6):200–214, 2012.

[4] P. B. Callahan and S. R. Kosaraju Faster algorithms for some geometric graph problems in higher dimensions. *In the Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*:291–300, 1993.

[5] J. Fischer and S. Har-Peled Dynamic well-separated pair decomposition made easy. *CCCG05: Proceedings of the 17th Canadian Conference on Computational Geometry*:235–238, 2005.

Spanners on imprecise points

Abolfazl Poureidi*† Mohammad Farshi†

Abstract

In this paper, we provide an imprecise point set R of n straight-line segments in the plane such that any imprecise t-spanner for R has $\Omega(n^2)$ edges. Then, we give an algorithm that computes an imprecise t-spanner for a set of n pairwise disjoint d-dimensional balls with arbitrary sizes. This imprecise t-spanner has $\mathcal{O}(n/(t-1)^d)$ edges and can be computed in $\mathcal{O}(n \log n/(t-1)^d)$ time.

1 Introduction

A *geometric network* is a weighted undirected graph whose vertices are points in $\mathbb{R}^d$, and in which each edge is a straight-line segment with weight equal to the Euclidean distance between its endpoints. In a geometric network $G = (P, E)$ on a set P of n points, the *graph distance* of $u, v \in P$, denoted by $d_G(u, v)$, is the length of the shortest path between u and v in G. Then, $\delta_G(u, v) = \frac{d_G(u,v)}{|uv|}$ is the dilation between u and v in G. We say that there exists a *t-path* ($t > 1$) between two vertices $u, v \in P$ in G if $\delta_G(u, v) \leq t$ and a network G is called a *t-spanner* if $\delta_G(u, v) \leq t$ for any pair of distinct points $u, v \in P$.

We call any set $R = \{R_1, \ldots, R_n\}$ of n regions in $\mathbb{R}^d$ an *imprecise point* set. For a given imprecise point set R, any set $S = \{p_1, \ldots, p_n\}$, where $p_i \in R_i$ for each $1 \leq i \leq n$, is called a precise instance from R. For a given imprecise point set R, a graph $G = (R, E)$, where E is a set of unordered pairs of regions in R, is called an imprecise geometric graph. Given an imprecise geometric graph $G = (R, E)$, and for each precise instance S from R, we call the geometric graph $G_S = (S, E_S)$, where $E_S = \{(p_i, p_j) | (R_i, R_j) \in E\}$, a precise instance of G corresponding to S. Also, we call G an imprecise t-spanner ($t > 1$), if G_S, for all precise instances S from R, is a t-spanner.

Abam et al. [1] considered the problem of constructing a spanner for n pairwise disjoint balls in $\mathbb{R}^d$. For a given $t > 1$, Abam et al. [1] showed that there exists an imprecise t-spanner with $\mathcal{O}(n/(t-1)^d)$ edges that can be computed in $\mathcal{O}(n \log n/(t-1)^d)$ time when all balls have similar sizes. Their spanner construction was based on the Well-Separated Pair Decomposition (WSPD) [4] approach. When the balls sizes vary greatly, they used a Semi-Separated Pair Decomposition (SSPD) [2, 5] to solve the problem. Abam et al. [1] proved that there is an imprecise t-spanner with $\mathcal{O}(n \log n/(t-1)^d)$ edges that can be computed in $\mathcal{O}(n \log n/(t-1)^d)$ time.

Zeng and Gaoy [6] considered constructing an Euclidean spanner for n balls in $\mathbb{R}^d$ with radius r in two phases. In the first phase, they preprocessed balls in time $\mathcal{O}(n(r + 1/\varepsilon)^d \log \alpha)$, where α is the ratio between the farthest and the closest pair of centers of the balls. In the second phase, they could compute (or update) a $(1+\varepsilon)$-spanner for any precise instance of the balls with $\mathcal{O}(n(r+1/\varepsilon)^d)$ edges in time $\mathcal{O}(n(r+1/\varepsilon)^d \log(r+1/\varepsilon))$.

In Section 2 of this paper, we give a set of pairwise disjoint regions in the plane such that every imprecise t-spanner for the regions is the complete graph. Next, in Section 3 of this paper, we study the problem of computing an imprecise t-spanner for n pairwise disjoint balls in $\mathbb{R}^d$, given a real number $t > 1$. These balls have arbitrary sizes. We present an algorithm that computes an imprecise t-spanner with $\mathcal{O}(n)$ edges in $\mathcal{O}(n \log n)$ time, when t and d are constants.

2 An imprecise spanner with quadratic size

In this section, we present a set of pairwise disjoint convex regions in the plane such that any imprecise t-spanner for the regions, for any given $t > 1$, must be the complete graph. So, it is not interesting to study imprecise spanners for any set of regions.

Let $n \geq 2$ be an integer, and define $\theta := 2\pi/n$. If we rotate the positive x-axis by angles $i\theta$, for each i with $0 \leq i < n$, then we get n rays. We number the rays starting from the positive x-axis and in the counterclockwise direction. We denote the set of all these rays by $\mathcal{R}_n^o$.

Let us model an imprecise point as a line segment, and let O_n^t be the set of pairwise disjoint line segments in the plane that is constructed as follows. Let D_1 and D_2 be two disks centered at the origin with radii 0.4 and $(t + 1)/2$, respectively. Let p_i and q_i, for each $0 \leq i < n$, be the intersections of the i-th ray in $\mathcal{R}_n^o$ with the boundaries of D_1 and D_2, respectively. The line segment between p_i and q_i, denoted by (p_i, q_i), is an element of O_n^t, see Fig. 1. We have $|p_i q_i| > t/2$.

*Faculty of Mathematical Sciences, Shahrood University of Technology, Shahrood, Iran, a.poureidi@shahroodut.ac.ir

†Combinatorial and Geometric Algorithms Lab., Department of Mathematical Sciences, Yazd University, Yazd, Iran, mfarshi@yazd.ac.ir

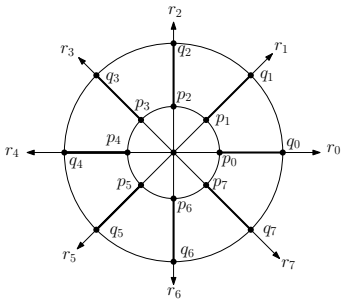


Figure 1: Illustrating O_8^t.

Lemma 1 *Every imprecise t-spanner of O_n^t is a complete graph for each $t > 1$.*

Proof. Assume that (p_i, q_i) and (p_j, q_j) are two distinct line segments in O_n^t, where $0 \leq i < j < n-1$. Let G be an imprecise t-spanner for O_n^t with no edge between (p_i, q_i) and (p_j, q_j). Consider the precise instance $S = \{q_0, \ldots, q_{i-1}, p_i, q_{i+1}, \ldots, q_{j-1}, p_j, q_{j+1}, \ldots, q_{n-1}\}$ from O_n^t, that is, choose p_i and p_j on (p_i, q_i) and (p_j, q_j), respectively, and q_k on other line segments. It is clear that $|p_i p_j| < 1$. Since there is no edge between p_i and p_j in G_S, the shortest path between p_i and p_j in G_S passes through some q_k. The Euclidean distance between p_i and q_k and the Euclidean distance between p_j and q_k are at least $t/2$ and, hence, it follows that $d_{G_S}(p_i, p_j) \geq |p_i q_k| + |q_k p_j| > t$. Therefore, we get $\delta_{G_S}(p_i, p_j) > t$, which is a contradiction, because we assume that G is an imprecise t-spanner for O_n^t. □

3 An imprecise spanner for balls

Let $D = \{D_1, \ldots, D_n\}$ be a set of n pairwise disjoint d-dimensional balls with arbitrary radii. In this section, we present an algorithm that computes an imprecise spanner for D with $\mathcal{O}(n)$ edges in $\mathcal{O}(n \log n)$ time. The algorithm uses the WSPD [4, 3] for computing the imprecise spanner. For our purpose we need the following.

Let X be a bounded subset of $\mathbb{R}^d$. We define the bounding rectangle of X, denoted by $R(X)$, as the smallest axes-parallel d-dimensional hyperrectangle that contains X. A d-dimensional hyperrectangle R is the Cartesian product of d closed intervals. More formally, $R = [l_1, r_1] \times [l_2, r_2] \times \cdots \times [l_d, r_d]$, where l_i and r_i are real numbers with $l_i \leq r_i$, for $1 \leq i \leq d$. We denote the length of R in the i-th dimension by $L_i(R) = r_i - l_i$. We denote the maximum and minimum lengths of R by $L_{\max}(R)$ and $L_{\min}(R)$, respectively. Let C_X be a d-dimensional ball that contains $R(X)$. We denote the distance between two disjoint d-dimensional balls C and C' by $d(C, C')$, i.e., $d(C, C') = |cc'| - (r + r')$, where c and r are, respectively, the center and radius of C, and c' and r' are the center and radius of C'. Clearly, if C or C' is a point, then its radius is zero.

Definition 1 *[4, 3] Let $s > 0$ be a real number, and let A and B be two finite sets of points in $\mathbb{R}^d$. We say that A and B are well-separated with respect to s if there are two disjoint d-dimensional balls C_A and C_B, such that (i) C_A and C_B have the same radius, and (ii) $d(C_A, C_B) \geq s \times radius(C_A)$.*

In the following, we define an s-well-separated pair for sets A and B of balls. Assume that A or B contains at least one nondegenerate ball, i.e., a ball with a positive radius. Let $D = \{D_1, \ldots, D_n\}$ be a set of n pairwise disjoint d-dimensional balls with arbitrary sizes, and let c_i be the center of D_i, for $1 \leq i \leq n$. For any $A \subseteq D$, let $A' = \{c_i | D_i \in A\}$.

Definition 2 *Let $s > 0$ be a real number, and let A and B be two nonempty subsets of D. We say that A and B are well-separated with respect to s if there are two disjoint d-dimensional balls $C_{A'}$ and $C_{B'}$ with the same radius, such that one of the following conditions hold:*

- $|A| = |B| = 1$,
- $A = \{D_k\}$, *for some* $1 \leq k \leq n$, $|B| > 1$, *and* $d(c_k, C_{B'}) - r_k \geq (3s+4) \times radius(C_{B'})$,
- $|A| > 1$, $B = \{D_k\}$ *for some* $1 \leq k \leq n$, *and* $d(c_k, C_{A'}) - r_k \geq (3s+4) \times radius(C_{A'})$, *or*
- $|A| > 1$, $|B| > 1$, *and* $d(C_{A'}, C_{B'}) \geq (3s+4) \times radius(C_{A'})$.

Note that if all balls in A and B are degenerate (i.e., balls with radius 0 or equivalently points), Definitions 1 and 2 are equivalent. Let $S = \{p_1, \ldots, p_n\}$, where $p_i \in D_i$ for each i with $1 \leq i \leq n$, be a precise instance from D, and for any $A \subseteq D$, let $A_S = \{p_i \in S | D_i \in A\}$.

Lemma 2 *Let A and B be two nonempty subsets of D that are a well-separated pair with respect to s, where $s > 0$ is a real number and let $S = \{p_1, \ldots, p_n\}$ be an arbitrary precise instance from D. Then, $\{A, B\}$ is an s-well-separated pair.*

Proof. Refer to Appendix. □

Definition 3 *A well-separated pair decomposition (WSPD) for D, with respect to $s > 0$, is a set $\{\{A_1, B_1\}, \{A_2, B_2\}, \ldots, \{A_m, B_m\}\}$ of pairs of nonempty subsets of D, for some integer m, such that*

1. *for any i with $1 \leq i \leq m$, A_i and B_i are a s-well-separated pair (by Definition 2), and*
2. *for any two distinct balls D_p and D_q of D, where $1 \leq p, q \leq n$, there is a unique index i with $1 \leq i \leq m$, such that $D_p \in A_i$ and $D_q \in B_i$, or $D_p \in B_i$ and $D_q \in A_i$.*

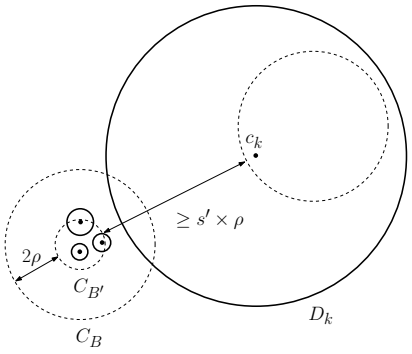


Figure 2: $A' = \{c_k\}$ and B', where $|B'| > 1$, are well-separated with respect to $s' = 3s + 4$, but $A = \{D_k\}$ and B are not s-well-separated.

Lemma 3 *Let $S = \{p_1, \ldots, p_n\}$ be an arbitrary precise instance from D. If $\{\{A_i, B_i\} | 1 \leq i \leq m\}$ is a WSPD for D with respect to s, then $\{\{A_{i_S}, B_{i_S}\} | 1 \leq i \leq m\}$ is a WSPD for $S = \{p_1, \ldots, p_n\}$ with respect to $s > 0$.*

Proof. By Lemma 2, the proof is straightforward. □

If we can compute a WSPD for D, then (by Lemma 3) we can compute a WSPD for any precise instance from D and, therefore, we can compute an imprecise spanner for D. Callahan, and Kosaraju [4] used the split tree to compute a WSPD for a point set in $\mathbb{R}^d$. We also use the split tree to compute a WSPD for D.

To compute a WSPD of D, we construct a split tree T on the centers of the balls in D. Then, we build a WSPD W' of the centers with respect to $3s + 4$ using T. Next, we transform W' to a WSPD of D, denoted by W, in the following way. For each pair $\{A', B'\}$ in W', if both A' and B' are singletons or both sets contain more than one element, then we add $\{A, B\}$ to W, where X is the set of all balls that their centers are in X'. Note that, by Lemma 2, A and B are well-separated with respect to s. Otherwise, one of the A' and B' is a singleton and the other one contains more than one element. In this case, it is possible that $\{A, B\}$ is not an s-well-separated pair, see Fig. 2. Without loss of generality, we assume that $|A'| = 1$ and $|B'| > 1$. We check the pair $\{A, B\}$ to see if it is an s-well-separated pair. If it is a well-separated pair, then we add it to W and otherwise we partition the set B' to $\{B'_i\}_i$ such that $\{A, B_i\}$ are s-well-separated pairs and then add them to W. For the details of the algorithm, see Algorithm 3.1.

In the following, we explain the details of the way of partitioning B'. For any node u of T, let S_u be the set of all points of S that are stored in the subtree of u. Let $\{A', B'\}$ be a pair of W' such that $A' = \{c_k\}$, for some $1 \leq k \leq n$, and $|B'| > 1$. Assume that v and w are the nodes of T such that $S_v = A'$ and $S_w = B'$. Obviously,

Algorithm 3.1: ComputeWSPD(D, s)

Input: $D = \{D_1, \ldots, D_n\}$ is a set of n balls in $\mathbb{R}^d$ with arbitrary sizes and s is a positive real number.

Output: A well-separated pair decomposition of D with respect to s.

```
F = {c_1, ..., c_n}, where c_i is the center of D_i;
T := split tree of F;
W' := a WSPD of F with respect to 3s + 4 using T;
W := ∅;
foreach {A', B'} ∈ W' do
    if (|A'| = 1 and |B'| = 1) or (|A'| > 1 and |B'| > 1) then
        Add {A, B} to W;
    else
        /* assume |A'| = 1 and |B'| > 1          */
        v := the leaf in T corresponding to A';
        w := the node in T corresponding to B';
        Add pairs generated by FindPairs(T, v, w) to W;
return W;
```

v is a leaf and w is an internal node of T. Since the bounding box of each node in the split tree is stored in the node, the bounding box of w (that is, $R(w)$) is stored at w. So, we can test in $\mathcal{O}(1)$ time whether there is a ball $C_{B'}$ containing B' such that $d(c_k, C_{B'}) - r_k \geq (3s + 4) \times \text{radius}(C_{B'})$. To this end, let $C_{B'}$ be the d-dimensional ball of the radius $(\sqrt{d}/2) \times L_{\max} R(w)$ centered at the center of $R(w)$, where $L_{\max} R(w)$ is the length of the longest side of $R(w)$. If $d(c_k, C_{B'}) - r_k \geq (3s+4) \times \text{radius}(C_{B'})$, then $\{A, B\}$ is an s-well-separated pair and so we add $\{A, B\}$ to W. Otherwise, we follow the above process by $\{v, w_l\}$ and $\{v, w_r\}$, where w_l and w_r are the left and the right children of w, respectively.

For details of the partition algorithm, denoted by FindPairs(T, v, w), see algorithm 3.2. We may assume without loss of generality that always $|S_v| = 1$, that is, v is a leaf of T. Clearly, the algorithm FindPairs(T, v, w) terminates. Now, we show that the algorithm generates a WSPD of D with $\mathcal{O}(n)$ pairs.

Lemma 4 *If $A' = \{c_k\}$, for some integer $1 \leq k \leq m$, and B', where $|B'| > 1$, are well-separated with respect to $3s + 4$, but $\{A, B\}$ are not a s-well-separated pair, then $r_k = \text{radius}(D_k) > \sqrt{d} \times L_{\max}(R(B'))$.*

Proof. Refer to Appendix. □

Lemma 5 *Set W is a WSPD for D with respect to s.*

Proof. It is easy to see that for all $\{A, B\} \in W$, $\{A, B\}$ is an s-well-separated pair. By [4], the proof of the second condition in Definition 3 is straightforward. □

Algorithm 3.2: FindPairs(T, v, w)

Input: An split tree T and a pair $\{v, w\}$, where v is a leaf and w is an internal node of the split tree T.

Output: A collection of well-separated pairs $\{A, B\}$ with respect to s, where $A' = S_v = \{c_k\}$ and $B' \subseteq S_w$.

```
if there is a ball C_{S_w} such that
d(c_k, C_{S_w}) − r_k ≥ (3s + 4) × radius(C_{S_w}) then
    return the pair {A, B}, where A' = S_v and B' = S_w;
else
    w_l := left child of w;
    w_r := right child of w;
    FindPairs(T, v, w_l);
    FindPairs(T, v, w_r);
```

It remains to prove an upper bound on $|W|$. We can partition the pairs in W into two categories. In the first category, there are pairs $\{A, B\}$ such that $\{A', B'\}$ is in W'. Since the size of W' is linear, obviously the number of pairs in this category is linear. The pairs in the second category are generated by partitioning the sets in pairs of W'. In the following lemma, we show that the number of pairs in this category is also linear. To this end, we show that any set B appears in at most a constant number of pairs in this category. Note that each pair in this category contains a singleton and a set containing at least one element.

Let Z be the set of all pairs of W' that FindPairs returns at least two pairs. More precisely, let

$$Z = \{\{A'_i, B'_i\} | 1 \leq i \leq q, \{A'_i, B'_i\} \in W', |A'_i| = 1, |B'_i| > 1\},$$

such that $S_v = A'_k$, for some leaf v of T, and $S_w = B'_k$, for some node w of T, and algorithm FindPairs(v, w) returns at least two pairs, for all k between 1 and q. Let $\{A_k, B\}$ be a pair returned by algorithm FindPairs(v, w) such that $B' = S_u \subset S_w$, for some node u of T. We represent each pair $\{A_k, B\}$ returned by an algorithm FindPairs as a directed pair (A_k, B). In the following, we apply a packing argument (similar to Lemma 9.4.3 of [3, Chapter 9]) to prove that each B is involved in at most a constant number (dependent only on s and d) of directed pairs. Let $\pi(u)$ be the parent of node u of T, except for the root. For the rest, we assume that B is a fixed set with the above description.

Lemma 6 *The set B involved in at most $(3s + 7)^d \times \Gamma(d/2 + 1)/\pi^{d/2}$ pairs in W, where Γ denotes Euler's gamma-function.*

Proof. Refer to Appendix. □

To sum-up, we have the following result.

Corollary 7 *The set W contains at most $\mathcal{O}(n)$ pairs.*

Lemma 5 and Corollary 7 immediately imply the following result.

Theorem 8 *Let $D = \{D_1, \ldots, D_n\}$ be a set of n d-dimensional pairwise disjoint balls with arbitrary sizes, and let $s > 0$ be a real number. There is a WSPD for D with respect to s of size $\mathcal{O}\left(\frac{s^d \times \Gamma(d/2+1)}{\pi^{d/2}} \times n\right)$. The WSPD can be computed in $\mathcal{O}\left(\frac{s^d \times \Gamma(d/2+1)}{\pi^{d/2}} \times n \log n\right)$ time using $\mathcal{O}\left(\frac{s^d \times \Gamma(d/2+1)}{\pi^{d/2}} \times n\right)$ space.*

Theorem 9 *Let $D = \{D_1, \ldots, D_n\}$ be a set of n pairwise disjoint balls in $\mathbb{R}^d$, and let $t > 1$ be a real number. There is an imprecise t-spanner for D with $\mathcal{O}(n/(t-1)^d)$ edges. This imprecise t-spanner can be computed in $\mathcal{O}(n \log n + n/(t-1)^d)$ time.*

Proof. Refer to Appendix. □

References

[1] M. A. Abam, P. Carmi, M. Farshi and M. Smid, On the power of the semi-separated pair decomposition, *Comput. Geom.* 46(6):631–639, 2013.

[2] M. A. Abam, M. de Berg, M. Farshi and J. Gudmundsson, Region-fault tolerant geometric spanners, *Discrete Comput. Geom.* 41(4):556–582, 2009.

[3] G. Narasimhan and M. Smid. Geometric spanner networks. Cambridge University Press, 2007.

[4] P. B. Callahan and S. R. Kosaraju, A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields, *J. ACM* 42(1):67–90, 1995.

[5] K. R. Varadarajan, A divide-and-conquer algorithm for min-cost perfect matching in the plane, In *Proceedings 39th Annual Symposium on Foundations of Computer Science* pages 320–329, 1998.

[6] J. Zeng and J. Gao. A linear time Euclidean spanner on imprecise points, In *CCCG* pages 118–124, 2014.

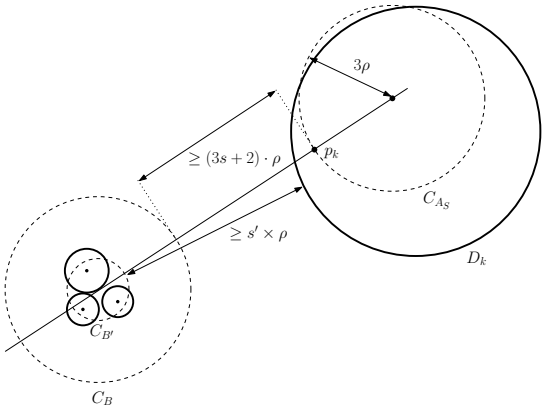


Figure 3: Illustrating C_{A_S} for $A = \{D_k\}$ and B, where $|B| > 1$, in the plane for the second case of Lemma 2.

Appendix

Lemma 2 Let A and B be two nonempty subsets of D that are well-separated with respect to s, where $s > 0$ is a real number and let $S = \{p_1, \ldots, p_n\}$ be an arbitrary precise instance from D. Then, $\{A, B\}$ is an s-well-separated pair.

Proof. Let $s' = 3s+4$. Recall that for any $A \subseteq D$, we have $A' = \{c_i | D_i \in A\}$, where c_i is the center of D_i. Since A and B are s-well-separated, by Definition 2, there are disjoint d-dimensional balls $C_{A'}$ and $C_{B'}$ with the same radius, such that one of the following cases holds for A and B. In each case, we prove that A_S and B_S are s-well-separated.

- $|A| = |B| = 1$. Since both A and B are singletons, it is clear that A_S and B_S are s-well-separated.
- $A = \{D_k\}$ for some $1 \leq k \leq n$, $|B| > 1$ and $d(c_k, C_{B'}) - r_k \geq s' \times \text{radius}(C_{B'})$. Let $\rho = \text{radius}(C_{B'})$, and C_B be a d-dimensional ball with radius 3ρ co-centered with $C_{B'}$. Since $|B| > 1$ and the balls in B are pairwise disjoint, the radius of each ball in B is at most 2ρ. (If B contains a ball with the radius greater than 2ρ, then B is a singleton, contradicting our assumption that $|B| > 1$.) Hence, C_B contains all balls in B and, as a result, C_B contains B_S. On the other hand, $d(c_k, C_B) - r_k = (d(c_k, C_{B'}) - 2\rho) - r_k \geq s' \times \rho - 2\rho = (3s+2) \times \rho$. Since A is a singleton, if we can pick the center of C_{A_S} on the line passing through p_k and the center of C_B such that p_k is on the boundary of C_{A_S} and p_k is between the centers of C_{A_S} and C_B, then d-dimensional ball C_{A_S} with radius 3ρ contains $p_k \in D_k$ such that its distance to C_B is at least $(3s+2) \times \rho$. See Fig. 3. It follows that

$$\begin{aligned} d(C_{A_S}, C_B) &\geq (3s+2)\rho \\ &\geq (3s)\rho \\ &= s \times (3\rho). \end{aligned}$$

  Therefore, A_S and B_S are a s-well-separated pair.
- $|A| > 1$, $B = \{D_k\}$ for some $1 \leq k \leq n$, and $d(c_k, C_{A'}) - r_k \geq s' \times \text{radius}(C_{A'})$. The proof is similar to the previous case.
- $|A| > 1$, $|B| > 1$, and $d(C_{A'}, C_{B'}) \geq s' \times \text{radius}(C_{A'})$. Let $\rho = \text{radius}(C_{A'}) = \text{radius}(C_{B'})$, and let C_A and C_B be two d-dimensional balls with radii 3ρ co-centered with $C_{A'}$ and $C_{B'}$, respectively. Hence, C_A contains all balls in A and, as a result, C_A contains all points of A_S, and C_B contains all balls in B and, as a result, C_B contains all points of B_S. The distance between C_A and C_B is at least $s' \times \rho - 4\rho$, i.e., $d(C_A, C_B) = d(C_{A'}, C_{B'}) - 4\rho \geq s'\rho - 4\rho = 3s\rho = s \times (3\rho)$. Therefore, A_S and B_S are s-well-separated.

This completes the proof. □

Lemma 4 If $A' = \{c_k\}$, for some integer $1 \leq k \leq m$, and B', where $|B'| > 1$, are well-separated with respect to $3s+4$, but A and B are not s-well-separated, then $r_k = \text{radius}(D_k) > \sqrt{d} \times L_{\max}(R(B'))$.

Proof. The proof is by contradiction. Assume that $r_k \leq \sqrt{d} \times L_{\max}(R(B'))$. Let $\rho = (\sqrt{d}/2) \times L_{\max}(R(B'))$. Since A' and B' are well-separated with respect to $3s+4$, we have $d(c_k, C_{B'}) \geq (3s+4) \times \rho$, where $C_{B'}$ is a ball with radius ρ that is centered at the center of $R(B')$. Let C_B is a ball with radius 3ρ co-centered with $C_{B'}$, and let C_A is a ball with radius 2ρ centered at c_k. Clearly, C_B contains all balls in B and C_A contains D_k. It follows that $d(C_A, C_B) \geq (3s) \times \rho$ and, therefore, A and B are s-well-separated, which is a contradiction. □

Lemma 6 The set B involved in at most $(3s+7)^d \times \Gamma(d/2+1)/\pi^{d/2}$ pairs in W, where Γ denotes Euler's gamma-function.

Proof. Let u be a node of T such that $S_u = B'$, and let $B'_p = S_{\pi(u)}$. Let x be the center of $R(S_{\pi(u)})$, and $\rho := \sqrt{d} \times L_{\max}(R(S_{\pi(u)}))$. Without loss of generality, we assume that $\{\{D_i\}, B\}_{1 \leq i \leq r} \in W$ are all pairs returned by algorithms FindPairs. Assume c_i and v_i are the center and the leaf of T corresponding to D_i, respectively.

Let C be a hypercube centered at x and with side length $(3s+5) \times \rho$. We have $C \cap D_i \neq \emptyset$, because if $C \cap D_i = \emptyset$, then $d(D_i, C_{B'_p}) = d(c_i, C_{B'_p}) - r_i > (3s+4) \times (\rho/2)$, where $C_{B'_p}$ is a ball with center x and radius $\rho/2$. (Clearly, $C_{B'_p}$ contains all points in B'_p.) Hence, $\{D_i\}$ and B_p are s-well-separated, which is a contradiction because if these two sets are s-well-separated, then FindPairs$(T, v_i, \pi(u))$ finishes and does not run FindPairs(T, v_i, u).

Since each element of Z is a well-separated pair with respect to $(3s+4)$, the pair $\{\{c_i\}, S_{\pi(u)}\}$ is also a well-separated pair with respect to $(3s+4)$. Since $\{D_i\}$ and B_p are not s-well-separated, by Lemma 4, for each i, $1 \leq i \leq r$, we have $r_i = \text{radius}(D_i) > \sqrt{d} \times L_{\max}(S_{\pi(u)}) = \rho$.

For each i, let C_i be a d-dimensional ball with radius ρ such that D_i contains C_i and $C \cap C_i \neq \emptyset$. Since, the balls D_i are pairwise disjoint, the balls C_i are also pairwise disjoint.

Let C' be a hypercube with sides of length $(3s+5)\rho + 2\rho$ and with center x. The length of sides of C' is the sum of the length of sides of C and two times the radius of C_i. Therefore, C' contains all balls C_i, for each i with $1 \leq i \leq r$. The volumes of $\bigcup_{i=1}^{r} C_i$ and C' are $r \times (\pi^{d/2}/\Gamma(d/2+1)) \times \rho^d$

and $((3s + 7) \times \rho)^d$, respectively. Therefore, we get $r \times (\pi^{d/2}/\Gamma(d/2 + 1)) \times \rho^d \leq ((3s + 7) \times \rho)^d$. It follows that

$$r \leq (3s + 7)^d \times \Gamma(d/2 + 1)/\pi^{d/2},$$

which completes the proof. □

Theorem 9 Let $D = \{D_1, \ldots, D_n\}$ be a set of n pairwise disjoint balls in $\mathbb{R}^d$, and let $t > 1$ be a real number. There is an imprecise t-spanner for D with $\mathcal{O}(n/(t-1)^d)$ edges. This imprecise t-spanner can be computed in $\mathcal{O}(n \log n + n/(t - 1)^d)$ time.

Proof. Let $s = 4(t + 1)/(t - 1)$ and, by Theorem 8, let $\{\{A_i, B_i\}|1 \leq i \leq m\}$ be a WSPD for D with respect to s of size $m = \mathcal{O}\left(\frac{s^d \times \Gamma(d/2+1)}{\pi^{d/2}} \times n\right)$. Initialize $E = \emptyset$. For $1 \leq i \leq m$, we add edge $\{D_j, D_k\}$ to E, where $D_j \in A_i$ and $D_k \in B_i$. Let $G = (D, E)$ be the resulting graph. By Theorem 8, G can be computed in $\mathcal{O}(n \log n)$ time. Let $S = \{p_1, \ldots, p_n\}$ be an arbitrary precise instance from D. By Lemma 3, $\{\{A_{i_S}, B_{i_S}\}|1 \leq i \leq m\}$ is a WSPD for S with respect to s. It follows from [4] that $G_S = (S, E_S)$ is a t-spanner for S, that is, $G = (D, E)$ is an imprecise t-spanner for D. □

On the Spanning Ratio of the Directed Θ_6-Graph*

Hugo A. Akitaya† Ahmad Biniaz‡ Prosenjit Bose†

Abstract

The family of Θ_k-graphs is an important class of sparse geometric spanners with a small spanning ratio. Although this class of geometric graphs is well-studied, no bound is known on the spanning and routing ratio of the directed Θ_6-graph. We show that the directed Θ_6-graph of a point set P, denoted $\overrightarrow{\Theta}_6(P)$, is a $\frac{14\sqrt{3}}{3}$-spanner.

1 Introduction

A geometric graph $G = (V, E)$ is a graph whose vertex set V is a set of points in the plane and whose edge set E are segments joining vertices. If the edges are weighted, typically, their weight is the Euclidean distance between their endpoints and we refer to such graphs as Euclidean geometric graphs. A spanning subgraph H of a weighted graph G is a t-spanner of G provided that the weight of the shortest path in H between any pair of vertices is at most t times the shortest path in G. The smallest constant t for which H is a t-spanner of G is known as the **spanning ratio** or the **stretch factor** of H.

There is a vast literature outlining different algorithms for constructing various geometric $(1 + \varepsilon)$-spanners of the complete Euclidean geometric graph (see [11, 16] for a survey of the field). One can view a t-spanner H of a graph G as an approximation of G. From this perspective, there are many parameters that can be used to measure how good the approximation is. The obvious parameter is the spanning ratio, however, many other parameters have been studied in addition to the spanning ratio such as the size, the weight, the maximum degree, connectivity, and diameter to name a few. The study of spanners is a rich subfield and many of the challenges stem from the fact that these parameters are sometimes opposed to each other. For example, a spanner with high connectivity cannot have low maximum degree. As such, many different construction methods have been proposed which outline trade-offs between the various parameters.

A geometric graph H being a $(1 + \varepsilon)$-spanner of the complete Euclidean geometric graph certifies the existence of a short path in H between every pair of vertices. Finding such a short path is as fundamental a problem as constructing a good spanner. Typically, most path-planning or routing algorithms are assumed to have access to the whole graph when computing a short path [10, 13, 15]. However, in many settings, the routing must be performed in an **online** manner. This setting presents different challenges since the whole graph is not available to the algorithm but the routing algorithm must explore the graph as it attempts to find a path. By providing the routing algorithm with a sufficient amount of memory or a large enough stream of random bits, one can successfully route online using a random walk [12, 17] or Depth-First Search [13]. The situation is more challenging if the online routing algorithm is to be **memoryless** and **local**, i.e. the only information available to the algorithm, prior to deciding which edge to follow out of the current vertex, is the coordinates of the current vertex, the coordinates of the vertices adjacent to the current vertex and the coordinates of the destination vertex. The main difficulty in designing these types of algorithms is that deterministic routing algorithms that are memoryless and local often fail by cycling [7].

Θ_k-graphs, introduced independently by Clarkson [9] and Keil and Gutwin [14], are an important class of $(1 + \varepsilon)$-spanners of the complete Euclidean geometric graph for $\varepsilon > 0$. Θ_k-graphs have bounded spanning ratio [2, 3, 4, 9, 14, 18] for all $k > 3$ and unbounded spanning ratio [1] for $k = 2, 3$. Informally, a Θ_k-graph is constructed in the following way: the plane around each vertex v is partitioned into k cones with apex v and cone angle $2\pi/k$. In each cone, v is joined to the point whose projection on the bisector of the cone is closest to v. Although this naturally gives rise to a directed graph (where the previously described edges are directed away from v), much of the literature on Θ_k-graphs has focused on the underlying undirected graph. For example, the tightest upper and lower bounds on the spanning ratio for Θ_k-graphs are proven on the underlying undirected graphs (see [4] for a survey). Given a planar point set P, to avoid any confusion, we will denote the directed version of the Θ_k-graph as $\overrightarrow{\Theta}_k(P)$ and the underlying undirected graph as $\Theta_k(P)$.

Note that the definition of $\overrightarrow{\Theta}_k$-graphs gives rise to a simple, online, local routing algorithm referred to as greedy Θ-routing: when searching for a path from a vertex s to a vertex d, follow the edge from s in the cone

*Research supported in part by NSERC.

†School of Computer Science, Carleton University, Ottawa, ON, Canada.

‡School of Computer Science, University of Windsor, Windsor, ON, Canada.

that contains d. Repeat this procedure until the destination is reached. At each step, the only information used to make the routing decision is the location of the destination and the edge out of the current vertex that contains the destination. Thus, greedy Θ-routing is online, local and memoryless. One main advantage of $\overrightarrow{\Theta}_k(P)$ over $\Theta_k(P)$ is that each vertex needs to store at most k outgoing edges whereas a linear number of edges may need to be stored in $\Theta_k(P)$. Ruppert and Seidel [18] showed that greedy Θ-routing has a routing ratio (ratio between the output path and the shortest path) of $1/(1-2\sin(\pi/k))$ for $k \geq 7$. Intuitively, it seems that the routing ratio should be worse than the spanning ratio for all values of k, since an online routing algorithm must explore the graph while searching for a short path. Indeed, this is true for all values of $k \geq 7$, except when k is a multiple of 4, in which case the routing ratio and the spanning ratio are $1+2\sin(\pi/k)/(\cos(\pi/k)-\sin(\pi/k))$, and this bound is tight in the worst case. No tight bounds are known for the spanning and routing ratios in $\overrightarrow{\Theta}_k$. For the current best known spanning and routing ratios for Θ_k-graphs, we refer the reader to [4]. For $3 < k < 7$, it was shown in [5] that greedy Θ-routing has unbounded routing ratio. Recently, it was shown that Θ_4 has bounded (nongreedy) routing ratio [3]. Although this is not claimed by the authors, a careful analysis of their proof concludes that their result actually carries over to the directed setting.

We focus on fundamental questions related to $\overrightarrow{\Theta}_6(P)$. Chew [8] showed that $\Theta_6(P)$ is a 2-spanner, which matches the lower bound. Chew's proof is constructive and can be converted to a nongreedy routing algorithm on the underlying undirected graph $\Theta_6(P)$. No bounds are known for the spanning and routing ratio for $\overrightarrow{\Theta}_6(P)$. All that is known is that it is strongly-connected [5]. We show that $\overrightarrow{\Theta}_6(P)$ is a $\frac{14\sqrt{3}}{3}$-spanner.

2 Preliminaries

A convex polygon C is **regular** if all its edges are of the same length. By $\|C\|$, we refer to the side length of C. The boundary of C is denoted as $bd(C)$ and the interior of C is denoted as $int(C)$. We call a triangle (resp. hexagon) **aligned** if each of its edges is parallel to a line of slope $\sqrt{3}$, slope 0 or slope $-\sqrt{3}$. Given two distinct points u, v in the plane, the canonical triangle of u with respect to v, denoted $\bigtriangledown_u^v$ is the regular aligned triangle where u is one of the vertices and v is on the edge of the triangle opposite u. Note that $\bigtriangledown_u^v$ is congruent to $\bigtriangledown_v^u$. Let $\langle\!\langle u \rangle\!\rangle^v$ be the regular aligned hexagon centered at u that has v on its boundary. The lines through u having slopes $\sqrt{3}$, slope 0 and slope $-\sqrt{3}$, respectively, partition the hexagon into 6 regular aligned triangles. Label these triangles $\bigtriangleup_{uv}^0, \ldots, \bigtriangleup_{uv}^5$ in counter-clockwise order with the convention that $\bigtriangleup_{uv}^0$ is the triangle below u with a horizontal base. When referring to these triangles or sets related to these triangles, indices are manipulated modulo 6. Note that $\bigtriangleup_{uv}^i$ for some $i \in \{0, \ldots, 5\}$ which has v on its base is identical to $\bigtriangledown_u^v$. This implies that $\left\|\bigtriangleup_{uv}^i\right\| = \|\bigtriangledown_u^v\| = \left\|\langle\!\langle u \rangle\!\rangle^v\right\|$. Finally, we note that a regular aligned hexagon defines a distance metric. Given two points u, v in the plane, the hexagonal distance between u and v, $d_{\hexagon}(u,v) = \left\|\langle\!\langle u \rangle\!\rangle^v\right\| = \left\|\langle\!\langle v \rangle\!\rangle^u\right\|$.

Given a set of points P in the plane, the directed Θ_6-graph whose vertex set is P is denoted $\overrightarrow{\Theta}_6(P)$. A directed edge (a,b) exists in $\overrightarrow{\Theta}_6(P)$ provided that $\bigtriangledown_a^b$ does not contain any point of $P \setminus \{a,b\}$. We make the following general position assumption on our point set P: no two points lie on a line of slope $\sqrt{3}$, slope 0 or slope $-\sqrt{3}$. Note that a slight rotation of the point set removes this, as such, this assumption does not take away from the generality of our results.

3 Upper Bound on the Spanning Ratio

Given a destination vertex $d \in \overrightarrow{\Theta}_6(P)$, we define the **greedy edge** of vertex v with respect to d to be the outgoing edge of v in $\bigtriangledown_v^d$. Recall that the routing strategy of repeatedly following the greedy edge at every step until the destination is reached is called **greedy routing**. The path found by the greedy routing algorithm is called the **greedy path**. Thus, the greedy path from s to d, denoted $\pi(s,d)$, is the path in $\overrightarrow{\Theta}_6(P)$ starting at s and where at every step, the greedy edge with respect to d is selected, until the destination d is reached.

Given a starting vertex s and a destination vertex d, by construction, we have that the canonical triangle, $\bigtriangledown_s^d$, is contained in the hexagon $\langle\!\langle d \rangle\!\rangle^s$. Let (s,a) be the first greedy edge in $\pi(s,d)$. Then, since a is in $\bigtriangledown_s^d$ we have that $d_{\hexagon}(a,d) < d_{\hexagon}(s,d)$. The inequality is strict since by our general position assumption a is contained in $int(\bigtriangledown_s^d)$. Therefore, at every step of the greedy routing algorithm, the hexagonal distance to the destination decreases. Since there are a finite number of points in P and the fact that the hexagonal distance to the destination is strictly decreasing at every step, the greedy algorithm terminates at d. We summarize this in the following two lemmas.

Lemma 1 *Given any pair of points $s, d \in P$, there always exists a greedy path from s to d in $\overrightarrow{\Theta}_6(P)$.*

Lemma 2 *Let x be a vertex in $\pi(s,d)$ different from s and d. Then the following hold:*

- $\langle\!\langle d \rangle\!\rangle^x$ *is contained in* $int(\langle\!\langle d \rangle\!\rangle^s)$
- $d_{\hexagon}(x,d) < d_{\hexagon}(s,d)$
- $\pi(x,d)$ *is contained in* $\langle\!\langle d \rangle\!\rangle^x$

Although the greedy routing algorithm always reaches its destination, the spanning ratio of $\pi(s,d)$ is not bounded by a constant in the worst case [5]. The issue is that $\pi(s,d)$, although getting closer to d with respect to the hexagonal distance, can spiral around d many times. It is this observation that sparked our research.

We note that in the undirected version of $\overrightarrow{\Theta_6}(P)$, Chew [8] showed that there exists a path with spanning ratio at most 2 between any pair of points. As noted above, the reason that $\pi(s,d)$ can have unbounded spanning ratio is that the path can spiral around d many times. However, if there happens to be an edge from d to s, i.e. $(d,s) \in \overrightarrow{\Theta_6}(P)$, then $\pi(s,d)$ can no longer spiral around d since $\bigtriangledown_d^s$ is empty of points of P and acts as a barrier. This prevents the path from cutting across $\bigtriangledown_d^s$. We now prove that if (d,s) is an edge of $\overrightarrow{\Theta_6}(P)$ then the spanning ratio of $\pi(s,d)$ is at most $6\left\|\bigtriangledown_d^s\right\|$.

For $i \in \{0,\ldots,5\}$, let $T_i = \{(a,b) \in \pi(s,d) \mid a \in \triangle_{ds}^i\}$. T_i is the set of all edges of $\pi(s,d)$ whose tail is in $\triangle^i$. Define the weight of T_i, denoted $\|T_i\|$, to be $\sum_{(a,b)\in T_i}\left\|\bigtriangledown_a^b\right\|$. We show that the weight of T_i is an upper bound on the sum of the lengths of all the edges of $\pi(s,d)$ whose tails are in $\triangle^i$. This, in turn, allows us to bound the length of $\pi(s,d)$ since the tail of every edge of the greedy path is in one of the 6 triangles. We begin by showing that if (a,b) is an edge in T_i, then b can only be in $\triangle_{ds}^{i-1}, \triangle_{ds}^i$ or $\triangle_{ds}^{i+1}$. For ease of reference, label the sequence of vertices in $\pi(s,d)$ as $s = u_0,\ldots,u_k = d$ where k is the number of edges in the greedy path.

Lemma 3 *If (u_a,u_{a+1}) is an edge of $\pi(s,d)$ in T_i for $a \in \{0,\ldots,k-1\}$ and $i \in \{0,\ldots,5\}$, then u_{a+1} can only be in one of $\triangle_{ds}^{i-1}, \triangle_{ds}^i$ or $\triangle_{ds}^{i+1}$.*

The proof is given in the Appendix. Note that Lemma 3 immediately implies that the greedy path cannot spiral around d. Suppose that $int(\triangle_{ds}^3)$ is empty of points of P, i.e. (d,s) is an edge of $\overrightarrow{\Theta_6}(P)$, then there is no edge of $\pi(s,d)$ that has its head in $\triangle^2$ and its tail in $\triangle^4$ or vice versa. Thus, the greedy path cannot cross over $\triangle^3$. This property lets us bound the length of $\pi(s,d)$.

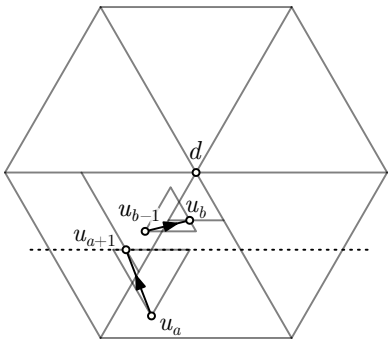


Figure 1: Illustration of Lemma 4.

Lemma 4 *Assume (d,s) is an edge of $\overrightarrow{\Theta_6}(P)$ and let u_a be a vertex of $\pi(s,d)$ in $\triangle_{ds}^i$. Let u_b be the next vertex in $\pi(s,d)$ after u_a that appears in $\triangle_{ds}^i$, i.e. $b > a$. Then, $int(\bigtriangledown_{u_a}^{u_{a+1}}) \cap int(\bigtriangledown_d^{u_b}) = \emptyset$.*

Proof. W.l.o.g., assume that u_a is in $\triangle_{ds}^0$. We have two cases to consider. Either $u_b = u_{a+1}$ or $u_b \neq u_{a+1}$. We begin with the former. If $u_b = u_{a+1}$ then the lemma holds trivially since u_{a+1} is on the horizontal edge of $\bigtriangledown_{u_a}^{u_{a+1}}$ and on the horizontal edge of $\bigtriangledown_d^{u_{a+1}}$.

Refer to Fig. 1. We now consider the case where u_b is not u_{a+1}, i.e. $b > a+1$. In this case, u_{a+1} must either be in $\triangle^1$ or $\triangle^5$ since u_b is the first vertex of $\pi(s,d)$ after u_a that is in $\triangle^0$. W.l.o.g., assume that u_{a+1} is in $\triangle^5$. Consider the edge (u_{b-1},u_b) of $\pi(s,d)$. By Lemma 3, u_{b-1} must be in $\triangle^5$ since, by the existence of (d,s), the path cannot spiral around d and enter $\triangle^0$ from $\triangle^1$. By Lemma 2, u_{b-1} must be contained in $\bigtriangledown_d^{u_{a+1}}$. Moreover, since (u_a,u_{a+1}) is an edge of the path, we have that $\bigtriangledown_{u_a}^{u_{a+1}}$ is empty, which means that u_{b-1} lies above the horizontal line through u_{a+1}. This implies that u_b also lies above the horizontal line through u_{a+1} since the canonical triangle $\bigtriangledown_{u_{b-1}}^{u_b}$ has a horizontal edge and lies above the horizontal line through u_{b-1}. Therefore, $int(\bigtriangledown_{u_a}^{u_{a+1}}) \cap int(\bigtriangledown_d^{u_b}) = \emptyset$. □

Lemma 5 *If $(d,s) \in \overrightarrow{\Theta_6}(P)$, then for each $T_i \subseteq \pi(s,d)$, we have $\|T_i\| \leq \|\bigtriangledown_d^s\|$, for $i \in \{0,\ldots,5\}$.*

We provide the proof in the Appendix. With Lemma 5 in hand, we are able to bound the length of $\pi(s,d)$ when $(d,s) \in \overrightarrow{\Theta_6}(P)$.

Lemma 6 *If $(d,s) \in \overrightarrow{\Theta_6}(P)$, then $\|\pi(s,d)\| \leq 6\|\bigtriangledown_d^s\| = 6\left\|\langle d\rangle^s\right\|$.*

Proof. Since each edge of $\pi(s,d)$ appears in only one T_i, the bound follows from Lemma 5. □

We note that Lemma 6 implies that $\overrightarrow{\Theta_6}(P)$ has a spanning ratio of 12. This follows from the fact that the underlying undirected graph has spanning ratio at most 2 and for each edge e in the underlying undirected graph there is a directed path of length at most $6\|e\|$ from one endpoint of e to the other in $\overrightarrow{\Theta_6}(P)$. A more careful analysis proves a better spanning ratio. To do so, we uncover a structural property of greedy paths in $\overrightarrow{\Theta_6}(P)$.

Lemma 7 *Between any pair of points $s,d \in P$, there exists an $x \in P$ in $\bigtriangledown_s^d$ such that the following hold (note that if the interior of $\bigtriangledown_s^d$ is empty then $x = d$):*

1. *$\pi(s,x)$ and $\pi(d,x)$ are both in $\bigtriangledown_s^d$,*
2. *$\|\pi(s,x)\| \leq \left\|\bigtriangledown_s^d\right\|$,*
3. *$\|\pi(d,x)\| \leq \left\|\bigtriangledown_s^d\right\|$.*

The proof is given in the Appendix and results from a careful analysis of the proof by Bose et al. [6] of the routing ratio of the half-Θ_6-graph. We now prove the main result of this section. We show that given any two points s, d in P, there exists a directed path $\delta(s,d)$ in $\overrightarrow{\Theta_6}(P)$ such that the length of $\delta(s,d)$ is at most $7\left\|\bigtriangledown_s^d\right\|$.

Theorem 8 *Between any pair of points $s, d \in P$, there exists a directed path $\delta(s,d)$ in $\overrightarrow{\Theta_6}(P)$ such that the length of $\delta(s,d)$ is at most $7\left\|\bigtriangledown_s^d\right\|$.*

Proof. Given a greedy path $\pi(a,b)$, the **reverse** path, denoted $\rho(b,a)$, is the path where every edge (x,y) in $\pi(a,b)$ is replaced with $\pi(y,x)$. Note that $\rho(b,a)$ is a directed path from b to a. By Lemma 6, $\|\rho(b,a)\| \leq 6\|\pi(a,b)\|$.

By Lemma 7, between any pair of points $s, d \in P$, there exists an $x \in \bigtriangledown_s^d$ such that $\pi(s,x)$ and $\pi(d,x)$ are both in $\overrightarrow{\Theta_6}(P)$, $\|\pi(s,x)\| \leq \left\|\bigtriangledown_s^d\right\|$, and $\|\pi(d,x)\| \leq \left\|\bigtriangledown_s^d\right\|$.

Let $\delta(s,d)$ be the path resulting from the concatenation of $\pi(s,x)$ and $\rho(x,d)$. By construction, $\delta(s,d)$ is a directed path from s to d.

$$\begin{aligned}\|\delta(s,d)\| &= \|\pi(s,x)\| + \|\rho(x,d)\| \\ &\leq \left\|\bigtriangledown_s^d\right\| + 6\left\|\bigtriangledown_s^d\right\| \text{ by Lemmas 6 and 7} \\ &= 7\left\|\bigtriangledown_s^d\right\|\end{aligned}$$

□

4 Conclusion

Since $\|sd\| > \frac{\sqrt{3}}{2}\left\|\bigtriangledown_s^d\right\|$, Theorem 8 implies a spanning ratio of $\frac{14\sqrt{3}}{3}$. Although the proof is constructive, unfortunately, it does not provide an online routing algorithm. There are three main obstacles. First, in the proof, the path is constructed from both ends, where we build a greedy path from s to x and another from d to x. Second, the point x is not easily identifiable locally. And third, when finding the reverse path of an edge (a,b), one needs to know both a and b. However, if we assume that each vertex in $\overrightarrow{\Theta_6}(P)$ is aware of only its outgoing edges, then finding the reverse path becomes problematic. We address these obstacles and present an online routing algorithm in an upcoming paper.

References

[1] O. Aichholzer, S. W. Bae, L. Barba, P. Bose, M. Korman, A. van Renssen, P. Taslakian, and S. Verdonschot. Theta-3 is connected. *Computational Geometry: Theory and Applications*, 47(9):910–917, 2014.

[2] L. Barba, P. Bose, J.-L. De Carufel, A. van Renssen, and S. Verdonschot. On the stretch factor of the theta-4 graph. In *Proceedings of the 13th Algorithms and Data Structures Symposium (WADS 2013)*, volume 8037 of *Lecture Notes in Computer Science*, pages 109–120, 2013.

[3] P. Bose, J. D. Carufel, D. Hill, and M. H. M. Smid. On the spanning and routing ratio of Theta-four. In *Symposium on Discrete Algorithms*, pages 2361–2370. SIAM, 2019.

[4] P. Bose, J. D. Carufel, P. Morin, A. van Renssen, and S. Verdonschot. Towards tight bounds on theta-graphs: More is not always better. *Theoretical Computer Science*, 616:70–93, 2016.

[5] P. Bose, J.-L. De Carufel, and O. Devillers. Expected complexity of routing in Θ_6 and half-Θ_6 graphs. *arXiv preprint arXiv:1910.14289*, 2019.

[6] P. Bose, R. Fagerberg, A. van Renssen, and S. Verdonschot. Optimal local routing on delaunay triangulations defined by empty equilateral triangles. *SIAM Journal on Computing*, 44(6):1626–1649, 2015.

[7] P. Bose and P. Morin. Online routing in triangulations. *SIAM Journal on Computing*, 33(4):937–951, 2004.

[8] P. Chew. There are planar graphs almost as good as the complete graph. *Journal of Computer and System Sciences*, 39(2):205–219, 1989.

[9] K. Clarkson. Approximation algorithms for shortest path motion planning. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing (STOC 1987)*, pages 56–65, 1987.

[10] E. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[11] D. Eppstein. Spanning trees and spanners. *Handbook of Computational Geometry*, pages 425–461, 1999.

[12] G. Grimmett. *Probability on Graphs: Random Processes on Graphs and Lattices: Second Edition.* Cambridge University Press, 2018.

[13] J. Hopcroft and R. Tarjan. Algorithm 447: Efficient algorithms for graph manipulation. *Communications of the ACM*, 16(6):372–378, 1973.

[14] J. M. Keil and C. A. Gutwin. Classes of graphs which approximate the complete Euclidean graph. *Discret. Comput. Geom.*, 7:13–28, 1992.

[15] C. Y. Lee. An algorithm for path connection and its applications. *IRE Transaction on Electronic Computers*, EC-10(3):346–365, 1961.

[16] G. Narasimhan and M. Smid. *Geometric Spanner Networks.* Cambridge University Press, 2007.

[17] K. Pearson. The problem of the random walk. *Nature*, 72:294, 1865.

[18] J. Ruppert and R. Seidel. Approximating the d-dimensional complete Euclidean graph. In *Proceedings of the 3rd Canadian Conference on Computational Geometry (CCCG 1991)*, pages 207–210, 1991.

Appendix

Submissions should not exceed four pages, must be submitted electronically, and must be prepared using LaTeX, this template.

Authors who feel that additional details are necessary should include a clearly marked appendix, which will be read at the discretion of the Program Committee.

Proof of Lemma 3.

Proof. W.l.o.g., assume that u_a is in $\triangle^0_{ds}$. Let $h^+(d)$ be the half-plane above the horizontal line through d. Since the edge of $\bigtriangledown^d_{u_a}$ that contains d is horizontal and the interior of the triangle lies below the horizontal line through d, we have that $int(\bigtriangledown^d_{u_a}) \cap h^+(d) = \emptyset$. Therefore, u_{a+1} cannot be in $\triangle^2, \triangle^3$ or $\triangle^4$, since the interiors of all those triangles are in $h^+(d)$. The lemma follows. □

Proof of Lemma 5

Proof. W.l.o.g., we show the bound for T_0. Let (a, b) be an edge in T_0. Let a' (resp. b') be the intersection of a horizontal line through a (resp. b) with the left side of $\triangle^0_{ds}$. Since $\bigtriangledown^b_a$ is equilateral, $\|a'b'\| \geq \|ab\|$. If (a, b) and (c, d) are in T_0, by Lemma 4, $a'b'$ and $c'd'$ do not overlap. Therefore, the sum of the lengths of all the edges in T_0 is at most $\|\bigtriangledown^s_d\|$. □

Proof of Lemma 7

Proof. The proof proceeds by induction on the rank of the canonical triangle $\bigtriangledown^d_s$ when the triangles are ordered by side length. We actually prove something slightly stronger. Consider an arbitrary pair of points $s, d \in P$. Refer to Fig. 2.

Notice that $\bigtriangledown^d_s$ can be decomposed into three regions. Let $C = \bigtriangledown^d_s \cap \bigtriangledown^s_d$ be the quadrilateral and let A and B be the two triangles that result from $\bigtriangledown^d_s \setminus \bigtriangledown^s_d$. Our inductive hypothesis states the following when the rank of $\left\|\bigtriangledown^d_s\right\|$ is at most k for $k \geq 1$:

1. If A does not contain any points of P, then there exists a point x in B such that $\|\pi(s, x)\| \leq \left\|\bigtriangledown^d_s\right\|$ and $\|\pi(d, x)\| \leq \|B\|$.
2. If B does not contain any points of P, then there exists a point x in A such that $\|\pi(s, x)\| \leq \left\|\bigtriangledown^d_s\right\|$ and $\|\pi(d, x)\| \leq \|A\|$.
3. If both A and B contain points of P, then then there exists a point x in $A \cup B$ such that $\|\pi(s, x)\| \leq \left\|\bigtriangledown^d_s\right\|$ and $\|\pi(d, x)\| \leq \max\{\|A\|, \|B\|\}$.

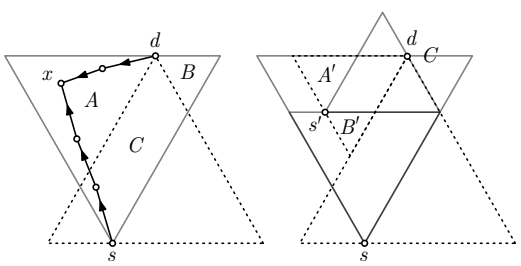


Figure 2: Illustrations for Lemma 7.

Base Case: (rank of $\left\|\bigtriangledown^d_s\right\|$ is 1): In this case, the theorem and the conditions of the inductive hypothesis trivially hold since the interior of $\bigtriangledown^d_s$ is empty of points of P, therefore, the edge (s, d) is in $\overrightarrow{\Theta_6}(P)$.

Inductive Step: We assume that the inductive hypothesis holds for all pairs of points where the length of the canonical triangle has rank at most k. Let $\left\|\bigtriangledown^d_s\right\|$ have rank $k+1$. Let (s, s') be the directed edge adjacent to s in $\bigtriangledown^d_s$. In this case, we consider the following 3 cases: s' is in C, s' is in A, and s' is in B. Since the last two cases are symmetric, we only address the first two.

Case 1: s' is in C. In this case, we need to address the three different subcases in the inductive hypothesis, namely A is empty of points, B is empty of points or neither is empty of points.

Case 1a: s' is in C and A is empty of points. Since s' is in $\bigtriangledown^d_s$, the rank of $\left\|\bigtriangledown^d_{s'}\right\|$ is at most k, which allows us to apply the inductive hypothesis. Let $C' = \bigtriangledown^d_{s'} \cap \bigtriangledown^{s'}_d$. Let $A' \subset A$ and $B' \subset B$ be the two triangles that result from $\bigtriangledown^d_{s'} \setminus \bigtriangledown^{s'}_d$. Since A is empty, A' is also empty. Therefore, by the inductive hypothesis there exists a point x in B' such that $\|\pi(s', x)\| \leq \left\|\bigtriangledown^d_{s'}\right\|$ and $\|\pi(d, x)\| \leq \|B'\|$.

We now need to show that $\|\pi(s, x)\| \leq \left\|\bigtriangledown^d_s\right\|$ and $\|\pi(d, x)\| \leq \|B\|$. Since $\|B'\| < \|B\|$, it follows that $\|\pi(d, x)\| \leq \|B\|$. Next we need to show that $\|\pi(s, x)\| \leq \left\|\bigtriangledown^d_s\right\|$. By construction (s, s') is the first edge in $\pi(s, x)$. Thus $\pi(s, x)$ is (s, s') followed by $\pi(s', x)$.

$$\|\pi(s, x)\| = \left\|(s, s')\right\| + \left\|\pi(s', x)\right\| \tag{1}$$
$$\leq \left\|\bigtriangledown^{s'}_s\right\| + \left\|\bigtriangledown^d_{s'}\right\| \tag{2}$$
$$\leq \left\|\bigtriangledown^d_s\right\| \tag{3}$$

We have that (2) follows from the inductive hypothesis and (3) follows since s' is in $\bigtriangledown^d_s$ and $int(\bigtriangledown^{s'}_s) \cap int(\bigtriangledown^d_{s'}) = \emptyset$.

Case 1b: s' is in C and B is empty of points. The argument is identical to Case 1a except that we swap the roles of B and B' with A and A', respectively.

Case 1c: s' is in C and neither A nor B is empty of points. The argument is fairly similar to the previous cases but we outline the differences. Label s', A', B' and C' as in Case 1a. The inductive hypothesis ensures the existence of a point x in $A' \cup B'$ such that $\|\pi(s', x)\| \leq \left\|\bigtriangledown^d_s\right\|$ and $\|\pi(d, x)\| \leq \max\{\|A'\|, \|B'\|\}$

We now need to show that $\|\pi(s, x)\| \leq \left\|\bigtriangledown^d_s\right\|$ and $\|\pi(d, x)\| \leq \max\{\|A\|, \|B\|\}$. Since $\max\{\|A'\|, \|B'\|\} < \max\{\|A\|, \|B\|\}$, it follows that $\|\pi(d, x)\| \leq \max\{\|A\|, \|B\|\}$. The inequality $\|\pi(s, x)\| \leq \left\|\bigtriangledown^d_s\right\|$ follows using the same argument outlined in Case 1a.

Case 2: s' is in A. In this case, we only need to address 2 subcases, namely when B is empty and when B is not empty.

Case 2a: s' is in A and B is empty of points. Since s' is in A, we focus on $\bigtriangledown^{s'}_d$. The rank of $\left\|\bigtriangledown^{s'}_d\right\|$ is at most k since s' is in A.

Let $C' = \bigtriangledown^{s'}_d \cap \bigtriangledown^d_{s'}$. Let A' and B' be the two triangles that result from $\bigtriangledown^{s'}_d \setminus \bigtriangledown^d_{s'}$. Let B' be the one that is contained in $\bigtriangledown^{s'}_s$. The key observation that allows us to

complete the proof is that B' is empty of points. This is because (s, s') is an edge which implies that $\bigtriangledown_s^{s'}$ is empty of points.

Therefore, by the inductive hypothesis there exists a point x in A' such that $\|\pi(d, x)\| \leq \left\|\bigtriangledown_d^{s'}\right\|$ and $\|\pi(s', x)\| \leq \|A'\|$.

We now need to show that $\|\pi(s, x)\| \leq \left\|\bigtriangledown_s^d\right\|$ and $\|\pi(d, x)\| \leq \|A\|$. Because s' is in A, we have that $\bigtriangledown_d^{s'}$ is contained in A. Thus, it follows that $\|\pi(d, x)\| \leq \|A\|$. Next we need to show that $\|\pi(s, x)\| \leq \left\|\bigtriangledown_s^d\right\|$. By construction (s, s') is the first edge in $\pi(s, x)$. Thus $\pi(s, x)$ is (s, s') followed by $\pi(s', x)$.

$$\|\pi(s, x)\| = \left\|(s, s')\right\| + \left\|\pi(s', x)\right\| \tag{4}$$

$$\leq \left\|\bigtriangledown_s^{s'}\right\| + \left\|A'\right\| \tag{5}$$

$$\leq \left\|\bigtriangledown_s^d\right\| \tag{6}$$

Case 2b: s' is in A and B is not empty of points. The argument is virtually identical to Case 2a and follows from the fact that $\max\{\|A'\|, \|B'\|\} < \max\{\|A\|, \|B\|\}$. □

The Minimum Moving Spanning Tree Problem*

Hugo A. Akitaya† Ahmad Biniaz‡ Prosenjit Bose† Jean-Lou De Carufel§ Anil Maheshwari†
Luís Fernando Schultz Xavier da Silveira† Michiel Smid†

Abstract

We investigate the problem of finding a spanning tree of a set of moving points in the plane that minimizes the maximum total weight (sum of Euclidean distances between edge endpoints) or the maximum bottleneck throughout the motion. The output is a single tree, i.e., it does not change combinatorially during the movement of the points. We call these trees the minimum moving spanning tree, and the minimum bottleneck moving spanning tree, respectively. We show that, although finding the minimum bottleneck moving spanning tree can be done in $O(n^2)$ time, it is NP-hard to compute the minimum moving spanning tree. We provide a simple $O(n^2)$-time 2-approximation and a $O(n \log n)$-time $(2+\varepsilon)$-approximation for the latter problem.

1 Introduction

The Euclidean minimum spanning tree (EMST) of a point set is the minimum weight graph that connects the given point set, where the weight of the graph is given by the sum of Euclidean distances between endpoints of edges. EMST is a classic data structure in computational geometry and it has found many uses in network design and in approximating NP-hard problems. In the visualization community, a series of methods generalize Euler diagrams to represent spatial data [6, 14]. These approaches represent a set by a connected colored shape containing the points in the plane that are in the given set. In order to reduce visual clutter, approaches such as Kelp Diagrams [6] and Colored Spanning Graphs [11] try to minimize the area (or "ink") of such colored shapes. Each of the shapes can be considered generalizations of the EMST of the points in the set.

Motivated by visualizations of time-varying spatial data, we investigate a natural generalization of the minimum spanning tree (MST) and the minimum bottleneck spanning tree (MBST) for a set of moving points. In general it is desirable that visualizations are stable, i.e., small changes in the input should produce small changes in the output [15]. In this paper, we want to maintain all points connected throughout the motion by the same tree (the tree does not change topologically during the time frame), thus obtaining a completely stable spanning tree. Consider points in the plane moving on a straight line with constant speed over a time interval $[0, 1]$. The weight of an edge pq between points p and q is defined to be the Euclidean distance $\|pq\|$. Note that the weight of an edge changes over time. We define the *minimum moving spanning tree* (MMST) of a set of moving points to be a spanning tree that minimizes the maximum sum of weights of its edges during the time interval. Analogously, we define *minimum bottleneck moving spanning tree* (MBMST) of a set of moving points to be a spanning tree that minimizes the maximum individual weight of edges in the tree during the time interval.

Apart from this motivation, the concepts of MMST and MBMST are relevant in the context of moving networks. Motivated by the increase in mobile data consumption, network architecture containing mobile nodes have been considered [12]. In this setting, the design of the topology of the networks is a challenge. Due to the mobility of the vertices, existing methods update the topology dynamically and the stability becomes important since there are costs associated with establishing new connections and handing over ongoing sessions. The MMST and MBMST again offer complete stability while minimizing a parameter.

Results and Organization. We study the problems of finding a MMST and a MBMST of a set of points moving linearly, each at constant speed. Section 2 provides formal definitions and proves that the distance function between points is convex in this setting. We use this property in an exact $O(n^2)$-time algorithm for the MBMST as shown in Section 3. Our algorithm computes the minimum bottleneck tree in a complete graph G between on the moving points in which the weight of each edge is the maximum distance between the pairs of points during the time frame. In Section 4 we present an $O(n^2)$-time 2-approximation for MMST by computing the MST of G. In the full version of the paper we provide an example that shows our analysis for the approximation ratio is tight. Also in the full version we show

*Research supported in part by NSERC.

†School of Computer Science, Carleton University, Ottawa, ON, Canada.
‡School of Computer Science, University of Windsor, Windsor, ON, Canada.
§School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, ON, Canada.

that the MMST is equal to the minimum spanning tree of a point set in $\mathbb{R}^4$ with a non-Euclidean metric. Since this metric space has doubling dimension $O(1)$, we obtain an $O(n \log n)$-time $(2+\varepsilon)$-approximation algorithm. In the full paper, we show that the MMST problem is NP-hard, by reducing from the Partition problem.

Related work. Examples of other visualizations of time-varying spatial data are space-time cubes [13], that represent varying 2D data points with a third dimension, and motion rugs [4], that reduces the dimentionality of the movement of data points to 1D, presenting a 2D static overview visualizations. The representation of time-varying geometric sets were also the theme of a recent Dagstuhl Seminar 19192 "Visual Analytics for Sets over Time and Space" [7]. In the context of algorithms dealing with time-varying data Meulemans et al. [15] introduces a metric for stability, analysing the trade-off between quality and stability of results, and applying it to the EMST of moving points. Monma and Suri [16] study the number of topological changes that occur in the EMST when one point is allowed to move.

The problem of finding the MMST and MBMST of moving points can be seen as a bicriteria optimization problem if the points move linearly (as shown in Section 2.2). In this context, the addition of a new criterion could lead to an NP-hard problem, such as the bi-criteria shortest path problem in weighted graphs. Garey and Johnson show that given a source and target vertices, minimizing both length and weight of a path from source to target is NP-hard [9, p. 214]. Arkin et al. analyse other criteria combined with the shortest path problem [2], such as the total turn length and different norms for path length.

Maintaining the EMST and other geometric structures of a set of moving points have been investigated by several papers since 1985 [3]. Kinetic data structures have been proposed to maintain the EMST [1]. Research in this area have focused on bounds on the number of combinatorial changes in the EMST and efficient algorithms. To the best of our knowledge, the problem of finding the MMST and MBMST (a single tree that does not change during the movement of points) has not been investigated.

2 Preliminaries

In this section we formally define the minimum moving spanning tree and the minimum bottleneck moving spanning tree of a set of moving points. We then prove that, for points moving linearly, the distance function between a pair of points is convex.

2.1 De nitions

A *moving point* p in the plane is a function $p : [0,1] \to \mathbb{R}^2$. We say that p is at $p(t)$ at time t. We are given a set $S = \{p_1, ..., p_n\}$ of moving points in the plane. A *moving spanning tree* T of S has S as its vertex set and weight function $w_T : [0,1] \to \mathbb{R}$ defined as $w_T(t) = \sum_{pq \in T} \|p(t)q(t)\|$. Let (S) denote the set of all moving spanning trees of S. Let $w(T) = \sup_t w_T(t)$ be the *weight of the moving spanning tree* T. A *minimum moving spanning tree* (MMST) of S is a moving spanning tree of S with minimum weight. In other words a MMST is in

$$\underset{T \in (S)}{\arg\min} \left(w(T)\right).$$

Let $b_T(t) = \sup_{pq \in T} \|p(t)q(t)\|$ denote the *bottleneck* of a tree T at time t. A *minimum bottleneck moving spanning tree* (MBMST) of S is a moving spanning tree of S that minimizes the bottleneck over all $t \in [0,1]$. In other words a MBMST is in

$$\underset{T \in (S)}{\arg\min} \left(\max_t b_T(t)\right).$$

2.2 Convexity

Let p and q be two moving points in the plane. We assume that these points move along (possibly different) lines at (possibly different) constant velocities. Thus, for any real number t, we can write the positions of p and q at time t as

$$p(t) = (a_p + u_p t, b_p + v_p t), \quad q(t) = (a_q + u_q t, b_q + v_q t),$$

where a_p, u_p, b_p, v_p are constants associated with the point p. At time $t = 0$, p is at (a_p, b_p), and the velocity vector of p is (u_p, v_p). Let $d(t) = \|p(t)q(t)\|$ denote the Euclidean distance between p and q at time t.

Lemma 1 *The function d is convex.*

Proof. It suffices to prove that the second derivative of d is non-negative for all real numbers t. We can write

$$d(t) = \sqrt{At^2 + Bt + C},$$

where A, B, and C depend only on $a_p, u_p, b_p, v_p, a_q, u_q, b_q$, and v_q. Observe that $A \geq 0$. Since $d(t)$ represents a distance, $At^2 + Bt + C \geq 0$ for all t in $\mathbb{R}$. It follows that the discriminant of this quadratic function is non-positive, i.e.,

$$B^2 - 4AC \leq 0. \tag{1}$$

Let $\alpha = -B/2A$ and $\quad = C/A - B^2/(4A^2)$. Then

$$d(t) = \sqrt{A} \cdot \sqrt{(t-\alpha)^2 + \quad}.$$

The second derivative of the function $\quad(t) = \sqrt{t^2 + \quad}$ is given by

$$''(t) = \frac{\quad}{(t^2 + \quad)^{3/2}}.$$

It follows from (1) that ≥ 0. Thus, $''(t) \geq 0$ for all t in $\mathbb{R}$. Since $d(t) = \sqrt{A} \cdot (t - \alpha)$, we have $d''(t) \geq 0$ for all t in $\mathbb{R}$, and in particular, for $t \in [0, 1]$. □

The convexity of the distance function (Lemma 1) implies the following corollary.

Corollary 2 *The largest distance between two moving points is attained either at the start or at the finish time.*

Let S be a set of n moving points in the plane. For two points p and q in S, we denote by $\|p(0)q(0)\|$ and $\|p(1)q(1)\|$ the distances between p and q at times $t = 0$ and $t = 1$, respectively. Moreover, we denote by $|pq|_{\max}$ the largest distance between p and q during time interval $[0, 1]$. By Corollary 2 we have

$$|pq|_{\max} = \max\{\|p(0)q(0)\|, \|p(1)q(1)\|\}. \qquad (2)$$

3 Minimum bottleneck moving spanning tree

Since by Corollary 2 the largest length of an edge is attained is either at time 0 or at time 1, it might be tempting to think that the MBMST of S is also attained at times 0 or 1. However the example in Figure 1(a) shows that this may not be true. In this example we have four points a, b, c, and d that move from time 0 to time 1 as depicted in the figure. The MBST of these points at time 0 is the red tree R, and their MBST at time 1 is the blue tree B. Recall that $b_T(t)$ is the bottleneck of tree T at time t. Let $b(T) = \max_t b_T(t)$ be the *bottleneck of a moving spanning tree* T. In R the weight of ab at time 0 is 1 while its weigh at time 1 is 3, and thus $b(R) = 3$. In B the weight of ad at time 1 is 1 while its weigh at time 0 is 3, and thus $b(B) = 3$. However, for this point set the tree $T = \{ac, cb, cd\}$ has bottleneck 2.

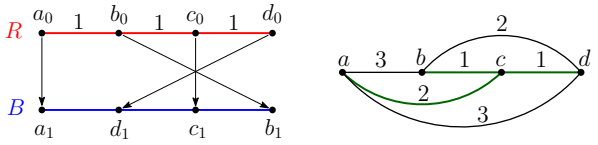


Figure 1: Four points moving from time 0 to 1. (a) R is the MBST at time 0, and B is the MBST at time 1. (b) The graph G; green edges form a MBST of this graph.

Although the above example shows that the computation of the MBMST is not straightforward, we present a simple algorithm for finding the MBMST. Let G be the complete graph on points of S where the weight $w(pq)$ of every edge pq is the largest distance between p and q during time interval $[0, 1]$, that is, $w(pq) = |pq|_{\max}$; see Figure 1(b).

Lemma 3 *The bottleneck of a MBMST of S is not smaller than the bottleneck of a MBST of G.*

Proof. Our proof is by contradiction. Let T^* be a MBMST of S and let T be a MBST of G. For the sake of contradiction assume that $b(T^*) < b(T)$, where we abuse the notation for simplicity making $b(T) = \max_{pq \in T} w(pq)$ the bottleneck of T. Let pq be a bottleneck edge of T, that is $b(T) = w(pq)$. Denote by T_p and T_q the two subtrees obtained by removing pq from T, and denote by V_p and V_q the vertex sets of these subtrees. Since the vertex set of T is the same as that of T^*, there is an edge, say rs, in T^* that connects a vertex of V_p to a vertex of V_q. Since the bottleneck of T^* is its largest edge-length in time interval $[0, 1]$, we have that $|rs|_{\max}$ $b(T^*)$. Since in G we have $w(rs) = |rs|_{\max}$, the following inequality is valid: $w(rs) = |rs|_{\max}$ $b(T^*) < b(T) = w(pq)$. Let T' be the spanning tree of G that is obtained by connecting T_p and T_q by rs. Then $b(T')$ $b(T^*)$. If we repeat this process for all bottleneck edges of T, then we obtain a tree T' whose bottleneck is strictly smaller than that of T. This contradicts the fact that T is a MBST of G. □

It is implied from Lemma 3 that any MBST of G is a MBMST of S. Since a MBST of a graph can be computed in time linear in the size of the graph [5], a MBST of G can be computed in $O(n^2)$ time. The following theorem summarizes our result in this section.

Theorem 4 *A minimum bottleneck moving spanning tree of n moving points in the plane can be computed in $O(n^2)$ time.*

4 Minimum moving spanning tree

In this section we present a 2-approximation algorithm for the problem. Our algorithm is simple and just computes a MST of the graph G constructed in Section 3.

Lemma 5 *The weight of any MST of G is at most two times the weight of any MMST of S.*

Proof. Let T be any MST of G and let T^* be any MMST of S. Let $w(T^*) = \sup_t w_T(t)$ be the *weight of the moving spanning tree* T^*. We abuse the notation for simplicity making $w(T) = \sum_{pq \in T} w(pq)$ the weight of the spanning tree T. We are going to show that $w(T)$ $2 \cdot w(T^*)$. Let T' be a tree that is combinatorially equivalent to T^*. Assign to each edge pq of T' the weight $w(pq) = |pq|_{\max}$. After this weight assignment, T' is a spanning tree of G. Since T is a MST of G, we have $w(T)$ $w(T')$.

By Corollary 2 the largest distance between two points is achieved either at time 0 or at time 1. Let E_0^* be the set of edges of T^* whose endpoints largest distance is achieved at time 0. Define E_1^* analogously.

Then $w(E_0^*)$ $w(T^*)$ and $w(E_1^*)$ $w(T^*)$. Moreover, $w(T') = w(E_0^*) + w(E_1^*)$. By combining these inequalities we get

$$w(T) \quad w(T') = w(E_0^*) + w(E_1^*) \\ w(T^*) + w(T^*) = 2 \cdot w(T^*).$$

□

A minimum spanning tree of G can be computed in $O(n^2)$ time using Prim's MST algorithm combined with a Fibonacci heap [8]. The following theorem summarizes our result in this section.

Theorem 6 *There is an $O(n^2)$-time 2-approximation algorithm for computing the minimum moving spanning tree of n moving points in the plane.*

In the full version of the paper we show that our analysis of the approximation factor is tight, as we build a set of moving points showing that the approximation factor of our 2-approximation algorithm can be arbitrarily close to 2. Moreover, we present an $O(n \log n)$-time $(2 + \varepsilon)$-approximation algorithm for the MMST problem. In the full version we also prove the NP-hardness of the MMST problem by a reduction from the Partition problem.

Acknowledgements

This research was carried out at the *8th Annual Workshop on Geometry and Graphs*, held at the Bellairs Research Institute in Barbados, Jan. 31–Feb. 7, 2020. The authors are grateful to the organizers and to the participants of this workshop. We would like to thank Günther Rote for pointing us to the work of Arkin et. al. [2].

References

[1] M. A. Abam, Z. Rahmati, and A. Zarei. Kinetic pie delaunay graph and its applications. In *Scandinavian Workshop on Algorithm Theory*, pages 48–58. Springer, 2012.

[2] E. M. Arkin, J. S. Mitchell, and C. D. Piatko. Bicriteria shortest path problems in the plane. In *Proc. 3rd CCCG*, pages 153–156, 1991.

[3] M. J. Atallah. Some dynamic computational geometry problems. *Computers & Mathematics with Applications*, 11(12):1171 – 1181, 1985.

[4] J. Buchmüller, D. Jäckle, E. Cakmak, U. Brandes, and D. A. Keim. Motionrugs: Visualizing collective trends in space and time. *IEEE Trans Vis Comput Graph*, 25(1):76–86, 2018.

[5] P. M. Camerini. The min-max spanning tree problem and some extensions. *Information Processing Letters*, 7(1):10–14, 1978.

[6] K. Dinkla, M. J. van Kreveld, B. Speckmann, and M. A. Westenberg. Kelp diagrams: Point set membership visualization. *Comput. Graph. Forum*, 31(3pt1):875–884, June 2012.

[7] S. I. Fabrikant, S. Miksch, and A. Wolff. Visual Analytics for Sets over Time and Space (Dagstuhl Seminar 19192). *Dagstuhl Reports*, 9(5):31–57, 2019.

[8] M. L. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM*, 34(3):596–615, 1987. Also in FOCS'84.

[9] M. R. Garey and D. S. Johnson. *Computers and intractability*, volume 174. freeman San Francisco, 1979.

[10] S. Har-Peled and M. Mendel. Fast construction of nets in low-dimensional metrics and their applications. *SIAM Journal on Computing*, 35(5):1148–1184, 2006.

[11] F. Hurtado, M. Korman, M. [van Kreveld], M. Löffler, V. Sacristán, A. Shioura, R. I. Silveira, B. Speckmann, and T. Tokuyama. Colored spanning graphs for set visualization. *Computational Geometry*, 68:262 – 276, 2018. Special Issue in Memory of Ferran Hurtado.

[12] S. Jaffry, R. Hussain, X. Gui, and S. F. Hasan. A comprehensive survey on moving networks. *arXiv preprint arXiv:2003.09979*, 2020.

[13] M.-J. Kraak. The space-time cube revisited from a geovisualization perspective. In *Proc. 21st International Cartographic Conference*, pages 1988–1996, 2003.

[14] W. Meulemans, N. H. Riche, B. Speckmann, B. Alper, and T. Dwyer. Kelpfusion: A hybrid set visualization technique. *IEEE Trans Vis Comput Graph*, 19(11):1846–1858, 2013.

[15] W. Meulemans, B. Speckmann, K. Verbeek, and J. Wulms. A framework for algorithm stability and its application to kinetic euclidean msts. In *Latin American Symposium on Theoretical Informatics*, pages 805–819. Springer, 2018.

[16] C. L. Monma and S. Suri. Transitions in geometric minimum spanning trees. *Discret. Comput. Geom.*, 8:265–293, 1992.

[17] G. Narasimhan and M. Smid. *Geometric Spanner Networks*. Cambridge University Press, Cambridge, UK, 2007.

Exploring the OT-graphs

Sergey Bereg* Mohammadreza Haghpanah*

Abstract

Recently, we introduced *OT-graphs* [4] for visualizing order types in the plane. The definition of OT-graphs uses abstract order types and their axioms described in the well-known book by Knuth [8]. Motivated by the problem of minimizing OT-graphs, we implemented an exhaustive search and explore the minimum OT-graphs for order types with a small number of points.

1 Introduction

Goodman and Pollack [7] introduced order types in 1983. Three points in the plane are either collinear, clockwise CW or counterclockwise CCW. In this paper we assume that point sets in the plane are in general position. Two finite point sets in the plane have the same *order type* if there is a bijection between them preserving orientation of any three distinct points. The equivalence classes defined by this equivalence relation are the *order types* [7].

Order type is a way of characterizing a finite point configuration's combinatorial properties. Order types are studied extensively. We refer readers to the survey on pseudoline arrangements [6] and the survey on oriented ma-troids [9].

1.1 Axioms and CC-systems

Knuth [8] introduced and studied *CC-systems* (short for "counterclockwise systems") axiomatizing order types. A *CC-system* for n points assigns true/false value for every ordered triple of points such that they satisfy the following axioms.

Axiom 1 (cyclic symmetry). $pqr \implies qrp$.

Axiom 2 (antisymmetry). $pqr \implies \neg prq$.

Axiom 3 (nondegeneracy). Either pqr or prq.

Axiom 4 (interiority). $tqr \wedge ptr \wedge pqt \implies pqr$.

Axiom 5 (transitivity). $tsp \wedge tsq \wedge tsr \wedge tpq \wedge tqr \implies tpr$.

Knuth [8] proved that Axioms 1,2,3, and 5 imply an axiom dual to Axiom 5.

Axiom 5' (dual transitivity). $stp \wedge stq \wedge str \wedge tpq \wedge tqr \implies tpr$.

*University of Texas at Dallas, Richardson, TX 75080, USA besp@utdallas.edu, Mohammadreza.Haghpanah@utdallas.edu

Consider the counterclockwise relation over triples of points in general position in the plane. It induces a CC-system. Due to the 9-point theorem of Pappus [5, 8], the converse is not valid. Knuth [8] proved that, given orientations of fewer than $\binom{n}{3}$ triples, it is NP-complete to decide whether there exists a CC-system extending them.

1.2 Visualization of order types

Recently, the problem of visualizing order types was studied in [2, 4] using *exit graphs* [2] and *OT-graphs* [4]. This problem is motivated by Aichholzer *et al.* [2] "... suppose we have discovered an interesting order type, and we would like to illustrate it in a publication." We also were facing this problem in [3] where we found that the order type 1874 for 9 points from the database [1] provides a (tight) lower bound for Tverberg partitions with tolerance 2, see Fig. 2(a). An order type in the plane can be represented by a corresponding point set (or explicit coordinates of the points). However, it might be difficult to recognize the orientations of some point triples.

1.3 OT-graphs

We define an OT-graph on a point set S as follows. Every edge (a, b) in an OT-graph is equipped with the partition of S by line ab, i.e. $S \setminus \{a, b\} = S^+_{ab} \cup S^-_{ab}$ where S^+_{ab} (S^-_{ab}) contains points $c \in S$ such that a, b, c has counterclockwise (clockwise) orientation. Using these orientations one can derive new orientations using the above axioms. An *OT-graph* contains a sufficient number of edges to decide the order type, i.e. to derive orientations of all triples of points.

It is easy to visualize the partitions of S for the edges of an OT-graph by drawing lines through them. This may result in a dense drawing, so we omit lines in the drawing if the partitions can be easily seen.

Example. Consider an order type for $n = 6$ and the graph G shown in Figure 1. There are $20 = \binom{6}{3}$ triples for $n = 6$ and 14 of them can be decided using graph G (i.e., 14 triples have two points which are endpoints of an edge). The remaining 6 triples are $(1, 3, 4)$, $(1, 3, 5)$,$(1, 3, 6)$, $(1, 4, 6)$, $(2, 4, 6)$ and $(3, 4, 6)$. We show that these triples can be decided using the axioms. By symmetry, it is sufficient to decide triples (1, 3, 4), (1, 3, 6), (2, 4, 6) and (3, 4, 6). Since $(1, 2)$ and $(2, 3)$ are

the edges of G, triples $(2,4,3)$, $(1,2,3)$, and $(1,4,2)$ are counterclockwise (CCW). By axiom 4 for $t=2, p=1, q=4, r=3$, triple $(1,4,3)$ is CCW. By axiom 4 for $t=2, p=1, q=6, r=3$, triple $(1,6,3)$ is CCW. By axiom 5 for $t=3, s=2, p=4, q=5, r=6$, triple $(3,4,6)$ is CCW. By axiom 5 for $t=2, s=1, p=4, q=5, r=6$, triple $(2,4,6)$ is CCW.

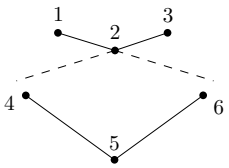


Figure 1: An OT-graph with 4 edges for an order type for $n=6$.

The second example is the OT-graph for the order type 1874 for 9 points from the database [1] shown in Fig. 2(b). The graph has 10 edges and it can be checked that it is an OT-graph.

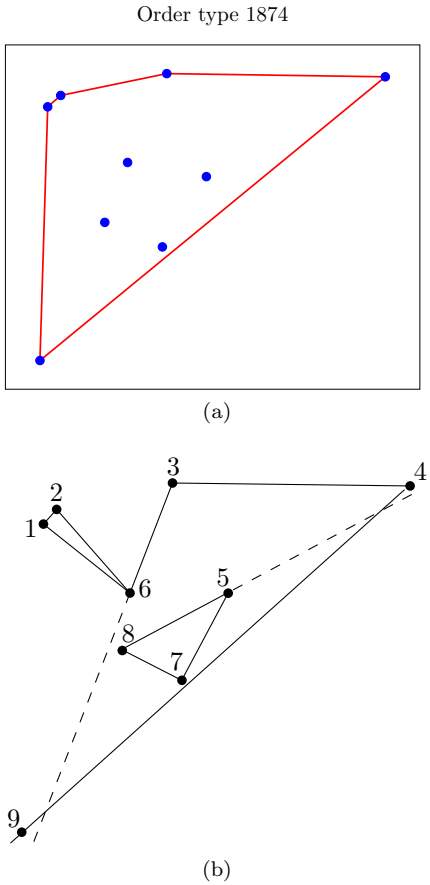


Figure 2: (a) The order type 1874 for 9 points from the database [1]. (b) An OT-graph with 9 edges for the order type 1874 (several OT-graphs with 9 edges were computed by an extensive search).

1.4 Minimum OT-graphs

Aichholzer *et al.* [2] suggested requirements for a graph representing an order type: "... we want to reduce the number of edges in the drawing as much as possible, but so that the order type remains uniquely identifiable." Let T be an order type. We use the algorithms from [4] to compute OT-graphs for T aiming to minimize the number of edges. It is difficult to find a *minimum OT-graph* for T (with large number of points), i.e. an OT-graph with minimum number of edges for T. In this paper (Section 3) we use exhaustive search for computing minimum OT-graphs. In Section 4, we discuss the results and conjectures.

2 OT-graphs for small point sets

In this section we present OT-graphs for point sets of size up to six. These graphs were computed using the greedy algorithm from [4]. In each step, the greedy algorithm chooses an edge (a,b) that maximizes the number of new triples in its CC-closure (for a set of triples with orientations, the *CC-closure* is the set of all triples that can be derived using the axioms from Section 1.1). In case of ties, the algorithm uses random tie-breaking.

For $n=3$, there is only one order type and a single edge can be used for an OT-graph. Figure 3 shows two OT-graphs for $n=4$ and three OT-graphs for $n=5$. Figure 4 shows 16 OT-graphs for $n=6$ which use point sets from from the database [1]. Note that all OT-graphs in Figures 3 and 4 are without crossings.

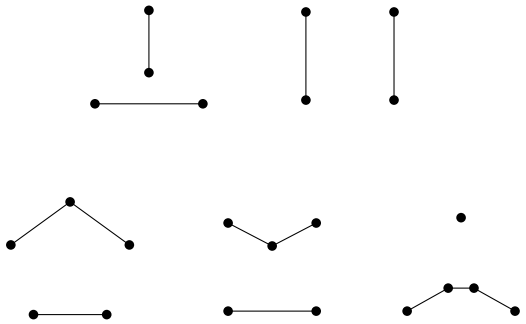


Figure 3: Order types for $n=4$ and $n=5$.

3 Minimum OT-graphs

In the previous section, we showed the OT-graphs for order types of up to six points constructed using the greedy algorithm. The greedy algorithm does not always find a *minimum OT-graph* for an order type, i.e. an OT-graph with minimum number of edges for the order type. Is it possible to verify that an OT-graph for an order type is minimum? One way is to prove lower bounds for the given order type using the axioms.

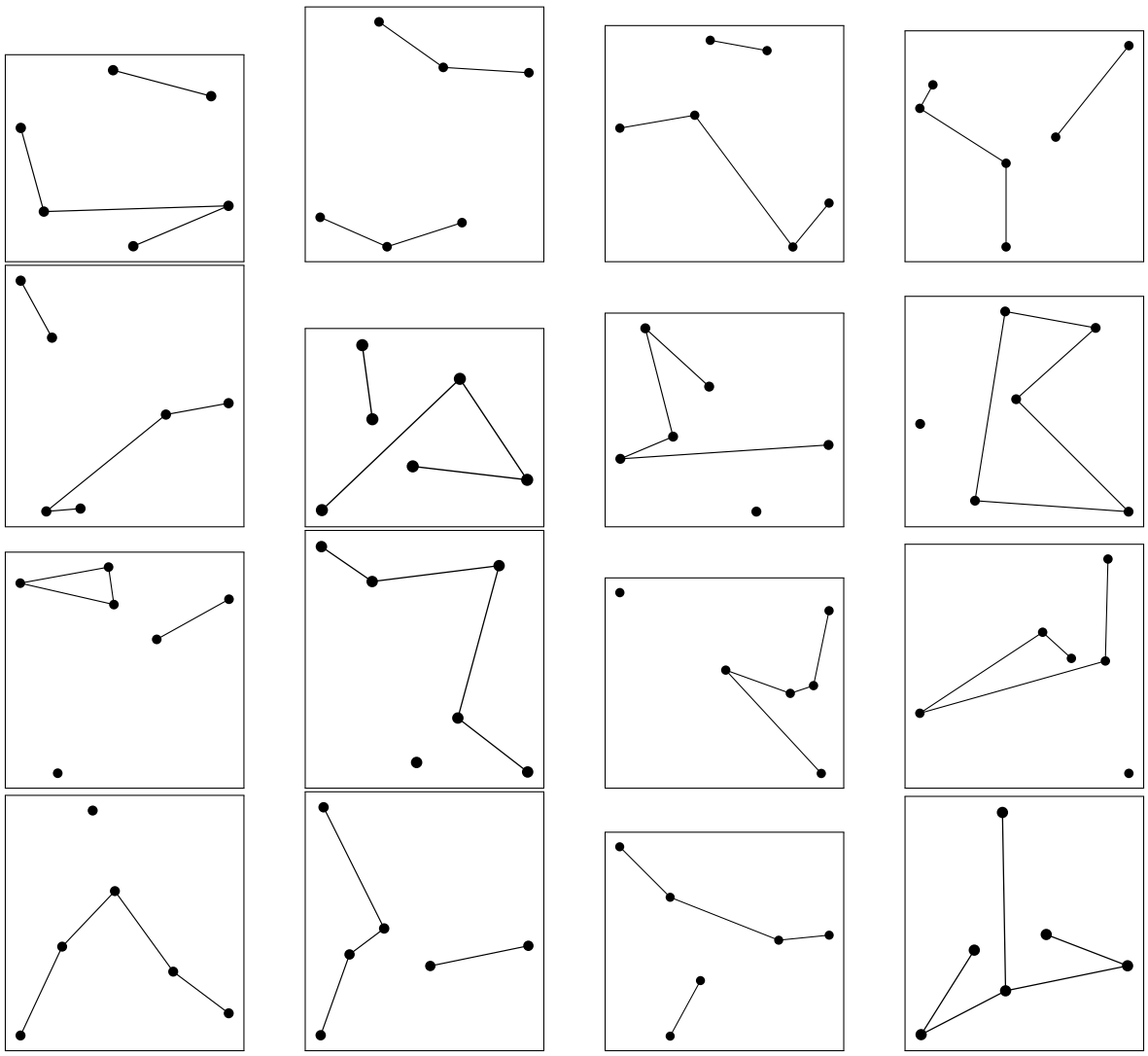


Figure 4: OT-graphs for 16 order types for $n = 6$.

This is challenging even for points in convex position [4]. In this paper we use computer aid to find minimum OT-graphs.

We used ComputingCC-Closure algorithm developed in [4]. Given an order type as point set and a graph G, the ComputingCC-Closure algorithm computes the CC-closure of G. The ComputingCC-Closure algorithm can be used for testing if a graph G is an OT-graph or not. We apply an exhaustive search using this testing algorithm. To prove that an OT-graph with m edges is minimum OT-graph, we check exhaustively every subset of $m-1$ edges in the complete graph.

We implemented the exhaustive search and check order types and OT-graphs shown in Figures 3 and 4. The program finds that all the OT-graphs in Figures 3 and 4 are minimum OT-graphs. The running time for all order type of up to six points was less than one seconds.

Next, we applied the exhaustive search for 135 order types of size 7. The execution time was around 10 minutes. The running time for 3,315 order types of size 8 increases significantly and the program is still running. For N OT-graphs with n vertices and m edges, the program tests

$$N \cdot \binom{\binom{n}{2}}{m-1}$$

graphs. This number grows as $2^{O(n^2)}$ even if we assume that N and m are constants. We expect the search for $n = 8$ to complete in one week.

Aichholzer *et al.* [1] provided the order types for $n < 11$ points publicly and for $n = 11$ upon request due to huge number of the order types. It is necessary to have more efficient search strategies to accomplish all the order types in the database [1].

Table 1: OT-graphs for n up to 9. Column $i, i = 1, 2, \ldots, 11$ contains the number of OT-graphs with i edges. The numbers shown in bold correspond to the minimum OT-graphs.

$n\backslash m$	1	2	3	4	5	6	7	8	9	10	11	total
3	**1**											1
4		**2**										2
5			**3**									3
6				**14**	**2**							16
7				**2**	**79**	**54**						135
8					59	1,061	1,716	479				3,315
9					5	890	15,235	54,304	67,011	21,145	227	158,817

4 Discussion

We computed OT-graphs using the greedy approach for the order types of size up to 10. The results are shown in Table 1. The order types of size n are shown in the corresponding row. The columns correspond to OT-graphs with m edges. For each order type, we count only one OT-graph with smallest number edges computed using the greedy algorithms.

The exhaustive search from the previous section was used to verify that the OT-graphs for n up to 7 are the minimum OT-graphs (and the numbers are shown in bold).

One edge less. Let T be an order type of size $n \leq 7$ and let m be the number of edges in the OT-graph for T. Recall that the exhaustive search checks every graph G with $m-1$ edges for T. Since m is the minimum, and the number of triples in CC-closure for G is less than $\binom{n}{3}$. It is interesting that sometimes the maximum number of triples in CC-closure for all graphs is one less than $\binom{n}{3}$. Therefore, these CC-closures are almost complete and we believe that, for these order types, it might be hard to prove that m is minimum. Figure 5 shows the minimum OT-graphs for $n = 6, 7$, and 8 of such examples.

Smallest graphs. Let $\mu(n)$ be the minimum number of edges in an OT-graph for n points. We conjectured in [4] that $\mu(4) = 2$, $\mu(5) = 3, \mu(6) = \mu(7) = 4, \mu(8) = \mu(9) = 5$. We confirm μ-values up to $n = 7$ using the exhaustive search.

Convex position. Let c_n be the minimum number of edges in an OT-graph for n points in convex position. The following upper bound for $c_n, n \geq 4$ is proved in [4] $c_n \leq \lfloor 2n/3 \rfloor$.

It is interesting to find exact values of sequence c_n. Using the exhaustive search, we found that $c_n = \lfloor 2n/3 \rfloor$ hold for $n = 4, 5, 6, 7$.

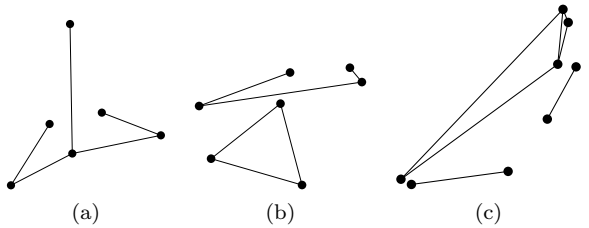


Figure 5: The minimum OT-graphs for some order types for $n = 6, 7$, and 8 such that there exists a graph with $m-1$ edges (one less) with CC-closure of size 19, 34, and 55 in (a),(b), and (c), respectively.

References

[1] O. Aichholzer, F. Aurenhammer, and H. Krasser. Enumerating order types for small point sets with applications. *Order*, 19(3):265–281, 2002.

[2] O. Aichholzer, M. Balko, M. Hoffmann, J. Kynčl, W. Mulzer, I. Parada, A. Pilz, M. Scheucher, P. Valtr, B. Vogtenhuber, and E. Welzl. Minimal representations of order types by geometric graphs. In *Graph Drawing*, pages 101–113, 2019.

[3] S. Bereg and M. Haghpanah. New lower bounds for Tverberg partitions with tolerance in the plane. *Discrete Applied Mathematics*, 283:596 – 603, 2020.

[4] S. Bereg and M. Haghpanah. Constructing order type graphs using an axiomatic approach. In *International Conference on Combinatorial Optimization and Applications*. Springer, to appear in Dec. 2020.

[5] J. Bokowski, J. Richter, and B. Sturmfels. Nonrealizability proofs in computational geometry. *Discrete & Computational Geometry*, 5(4):333–350, 1990.

[6] S. Felsner and J. E. Goodman. Pseudoline arrangements. In *Handbook of Discr. and Computat. Geom.*, pages 125–157. Chapman and Hall/CRC, 2017.

[7] J. E. Goodman and R. Pollack. Multidimensional sorting. *SIAM Journal on Computing*, 12(3):484–507, 1983.

[8] D. E. Knuth. *Axioms and Hulls*, volume 606 of *Lecture Notes in Computer Science*. Springer, 1992.

[9] J. Richter-Gebert and G. M. Ziegler. Oriented matroids. In *Handbook of Discr. and Computat. Geom.*, pages 159–184. Chapman and Hall/CRC, 2017.

A short proof of the non-biplanarity of K_9

Ahmad Biniaz*

Abstract

Battle, Harary, and Kodama (1962) and independently Tutte (1963) proved that the complete graph with nine vertices is not biplanar. Aiming towards simplicity and brevity, in this note we provide a short proof of this claim.

1 Introduction

An embedding (or drawing) of a graph in the Euclidean plane is a mapping of its vertices to distinct points in the plane and its edges to smooth curves between their corresponding vertices. A planar embedding of a graph is a drawing of the graph such that no two edges cross. A graph that admits such a drawing is called planar. A *biplanar embedding* of a graph $H = (V, E)$ is a decomposition of H into two planar graphs $H_1 = (V, E_1)$ and $H_2 = (V, E_2)$ such that $E_1 \cup E_2 = E$ and $E_1 \cap E_2 = \emptyset$, together with planar embeddings of H_1 and H_2. In this case, H is called *biplanar*. In other words, a graph is called biplanar if it is the union of two planar graphs; that is, if its thickness is 1 or 2. The *complete graph* with n vertices, denoted by $_n$, is a graph that has an edge between every pair of its vertices. Let G be a subgraph of $_n$ that has n vertices. The *complement* of G, denoted by $\overline{G}$, is the graph obtained by removing all edges of G from $_n$.

As early as 1960 it was known that $_8$ is biplanar and $_{11}$ is not biplanar. There exist several biplanar embeddings of $_8$; see e.g. [2] for a self-complementary drawing. The non-biplanarity of $_{11}$ is easily seen, since it has 55 edges while a planar graph with eleven vertices cannot have more than 27 edges, by Euler's formula. Finding the smallest integer n, for which $_n$ is non-biplanar, was a challenging question for some time [4]. The following fundamental theorem due to Battle, Harary, and Kodama ([1], 1962) and independently proved by Tutte ([8], 1963) answers this question and implies that is non-biplanar.

Theorem 1 *Every planar graph with at least nine vertices has a nonplanar complement.*

Both proofs of Theorem 1 involve a thorough case analysis. Battle, Harary, and Kodama gave an outline of a proof through six propositions. Some of these propositions require detailed case analysis, which are not given, for example (in their words) "There are several cases to discuss in order to establish Propositions 4 and 5. In each case, we can prove that $\overline{G}$ contains a subgraph homemorphic to $_{3,3}$ or $_5$." Tutte's proof appeared longer (it's a 13-page paper), and enumerates all simple triangulations (with no separating triangles) with up to 9 vertices and verifies that the complement of each triangulation is nonplanar. It seems that Harary was not quite satisfied with any of these proofs as he noted in his Graph Theory book [5] that "This result was proved by exhaustion; no elegant or even reasonable proof is known." We are still unaware of any short proof of this result. (See [6] for a recent attempt towards a new proof.)

2 Our proof

In this section we present a short proof of Theorem 1. Our proof is complete, self-contained, and only uses Kuratowski's theorem for non-planar graphs. Towards our proof we show (in Theorem 2) that a particularly restricted drawing of $_8$ cannot be biplanar; see Figure 1(a) for an illustration.

Theorem 2 *Let H be an embedded planar graph with eight vertices such that the boundary of its outerface is a 5-cycle and there are no edges between the three vertices that are not on the outerface. Then the complement of H is nonplanar.*

Proof of Theorem 1. Consider a planar graph G with nine vertices. For the sake of contradiction assume that its complement $\overline{G}$ is also planar. Fix a planar embedding of G and a planar embedding of $\overline{G}$. For convenience we use G and $\overline{G}$ for referring to planar graphs and to their planar embeddings. If there are two vertices in G that lie on the same face and are not connected by an edge, then we transfer the corresponding edge from $\overline{G}$ to G and connect the two vertices by a curve in that face. After this operation both G and $\overline{G}$ remain planar. Repeating this process converts G to a triangulation in which the boundary of every face (including the outerface) is a triangle (i.e. a 3-cycle). If all the three vertices on the outerface of G are of degrees at most 4, then the removal of these vertices results a graph with six vertices whose outerface is a 3-cycle (observe that otherwise the symmetric difference of this outerface and that of G is a

*School of Computer Science, University of Windsor, Windsor, Canada, ahmad.biniaz@gmail.com. Supported by NSERC.

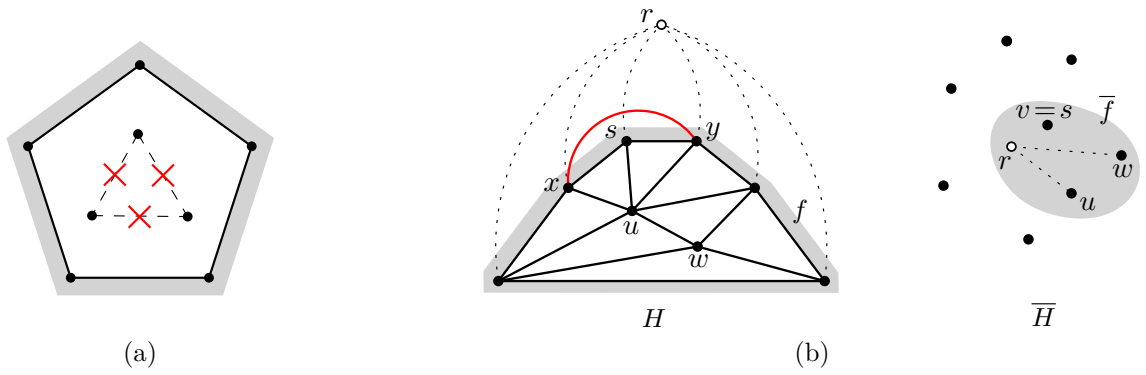


Figure 1: Illustrations of (a) the staement of Theorem 2 and (b) the proof of Theorem 1.

polygon, with a hole of size at least four, that requires at least seven edges to be triangulated). In this case the three vertices in the interior of this 3-cycle together with the three removed vertices form a $_{3,3}$ in $\overline{G}$ which contradicts its planarity. Thus we assume that at least one vertex, say r, on the outerface of G has degree k 5. Remove r from G and $\overline{G}$ and denote the resulting graphs by H and $\overline{H}$, respectively. Notice that $(H, \overline{H})$ is a biplanar embedding of $_8$. Let and $\overline{\ }$ be the faces of H and $\overline{H}$, respectively, that contain the removed vertex r, as in Figure 1(b). Observe that is the outerface of H. Each vertex of the resulting $_8$ lie on or on $\overline{\ }$ because $(G, \overline{G})$ was a biplanar embedding of . Since G was a triangulation, the outerface of H is a k-cycle. Since $\overline{G}$ was a simple graph (no multiedges and no loops), the face $\overline{\ }$ has at least three vertices; these vertices are not necessarily connected.

If $k > 5$ then let s be a vertex of that also lies on $\overline{\ }$; such a vertex exists because we have eight vertices in total. Let x and y be the neighbors of s on . If xy is an edge of H then draw it as a curve in . If xy is not an edge of H then transfer it from $\overline{H}$ to H and draw it in , as in Figure 1(b). Now, the new outerface of H has $k-1$ vertices. Repeat the above process until this outerface has exactly 5 vertices. Let u, v, w be the vertices of $_8$ that are not on this outerface. These three vertices lie on $\overline{\ }$ because of our choices of s. If any of the edges uv, uw, and vw is not in $\overline{H}$ then transfer it from H to $\overline{H}$ and draw it in $\overline{\ }$ without crossing other edges. To this end we obtained a planar graph H that satisfies the constraints of Theorem 2 and its complement $\overline{H}$ is planar. This contradicts Theorem 2. □

To prove Theorem 2 we use the well-known Kuratowski's theorem on non-planar graphs [3, 7] that "a finite graph is non-planar if and only if it contains a subgraph that is homeomorphic to $_5$ or $_{3,3}$." The following theorem, given in [8], is an alternate statement for Kuratowski's theorem.

Theorem 3 *A graph G is nonplanar if one of the following conditions hold: (i) G has six disjoint connected subgraphs $A_1, A_2, A_3, B_1, B_2, B_3$ such that for each A_i and B_j there is an edge edge with one end in A_i and the other in B_j. (ii) G has five disjoint connected subgraphs A_1, A_2, A_3, A_4, A_5 such that for each A_i and A_j, with $i \neq j$, there is an edge with one end in A_i and the other in A_j.*

Proof of Theorem 2. Let the 5-cycle $C = (a_1, a_2, a_3, a_4, a_5)$ be the boundary of the outerface of H, and let u, v, and w be the three vertices that are not on the outerface, i.e., lie on internal faces of H. For the sake of contradiction assume that the complement $\overline{H}$ of H is planar. By the statement of the theorem uv, uw, and vw are edges of $\overline{H}$. Except for the three pairs (u, v), (u, w), (v, w), if a pair of vertices lie on the same internal face of H and are not connected by an edge, then we transfer the corresponding edge from $\overline{H}$ to H and connect the two vertices by a curve in the face. After this operation both H and $\overline{H}$ remain planar. Repeating this process makes H edge-maximal (in the above sense).

Let H' be the embedded planar subgraph of H that is induced by the five vertices of C. The graph H' consists of the cycle C together with zero, one, or two chords as in Figure 2. Consider any internal face of H'. If contains some vertices of $\{u, v, w\}$ then by maximality of H it holds that exactly one of these vertices is connected to all boundary vertices of in H (see e.g. vertex v in Figures 2(a) and 2(b))—this holds as otherwise we could add an edge between two vertices of C or between a vertex of C and a vertex of $\{u, v, w\}$; contradicting the maximality of H. Now we consider three cases depending on the number of chords of H'. In each case we get a contradiction to planarity of $\overline{H}$.

- H' has no chords. Let v be the vertex of H that is connected to each a_i; see Figure 2(a). By planarity of H, each of u and w can only be adjacent to two consecutive vertices of C. Hence there exists a vertex of C (say a_1) that is adjacent to neither u nor w. Therefore, the five connected subgraphs u, w,

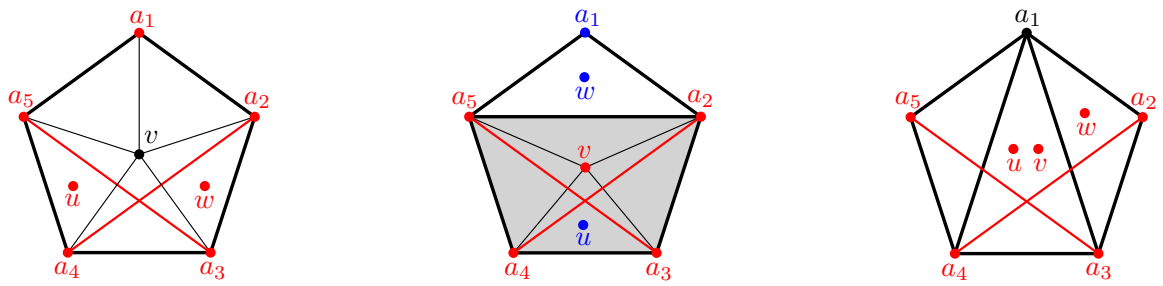


Figure 2: Black edges belong to H, bold black edges belong to H', and red edges belong to $\overline{H}$.

a_1, $\{a_2, a_4\}$ and $\{a_3, a_5\}$ from $\overline{H}$ satisfy condition (ii) of Theorem 3. Thus $\overline{H}$ is not planar.

- H' has one chord. After a suitable relabelling assume that this chord is (a_2, a_5). Let denote the face of H' whose boundary is the 4-cycle (a_2, a_3, a_4, a_5); this face is shaded in Figure 2(b). This face contains some vertices of $\{u, v, w\}$ because otherwise H' should have a chord in (by maximality of H) which contradicts our assumption that H' has one chord. Let v be the vertex in that is connected to all its boundary vertices. In this setting, regardless of where u and w lie, the six connected subgraphs u, w, a_1, v, $\{a_2, a_4\}$, and $\{a_3, a_5\}$ from $\overline{H}$ satisfy condition (i) of Theorem 3. Thus $\overline{H}$ is not planar.

- H' has two chords. Let a_1 be the vertex that is incident to the two chords as in Figure 2(c). In this setting, regardless of where u, v and w lie, the five connected subgraphs u, v, w, $\{a_2, a_4\}$, and $\{a_3, a_5\}$ from $\overline{H}$ satisfy condition (ii) of Theorem 3. Thus $\overline{H}$ is not planar.

□

References

[1] J. Battle, F. Harary, and Y. Kodama. Every planar graph with nine vertices has a nonplanar component. *Bulletin of the American Mathematical Society*, 68:569–571, 1962.

[2] L. Beineke. Biplanar graphs: A survey. *Computers & Mathematics with Applications*, 34(11):1–8, 1997.

[3] G. A. Dirac and S. Schuster. A theorem of Kuratowski. *Nederl. Akad. Wetensch. Proc. Ser. A*, 57:343–348, 1954.

[4] F. Harary. Problem 28. *Bulletin of the American Mathematical Society*, 67:542, 1961.

[5] F. Harary. *Graph theory*. Addison-Wesley, 1969.

[6] S. K. Kuila. Algebraic approach to prove non-coplanarity of . *International Journal of Engineering Inventions*, 4(6):19–23, 2014.

[7] K. Kuratowski. Sur le problème des courbes gauches en topologie. *Fundamenta Mathematicae*, 15:271–283, 1930.

[8] W. T. Tutte. On non-biplanar character of the complete 9-graph. *Canadian Mathematical Bulletin*, 6:319–330, 1963.

Spanning Trees in Geometric Graphs*

Anil Maheshwari†

This talk is primarily geared towards graduate students. We will discuss some recent results on spanning trees in geometric graphs. A geometric graph is a graph whose vertices are points in the plane and whose edges are straight-line segments between the points. The talk has three main parts.

In the first part, we will discuss the computation of minimum and maximum spanning trees for a bichromatic set of points. Let R and B be two disjoint sets of points in the plane where the points of R are colored red, and the points of B are colored blue, and let $n = |R \cup B|$. A bichromatic spanning tree is a spanning tree in the complete bipartite geometric graph with bipartition (R, B). The minimum (respectively maximum) bichromatic spanning tree problem is the problem of computing a bichromatic spanning tree of minimum (respectively maximum) total edge length. In the talk, we will discuss an algorithm running in $O(n \log^3 n)$ time for computing the required minimum and maximum spanning trees. We will also hint at how to obtain an optimal algorithm running in $\Theta(n \log n)$ time.

In the second part, we will discuss approximation algorithms for the computation of α-spanning trees of a point set. Let P be a set of points in the plane and let α be an angle. An α-ST of P is a spanning tree of the complete Euclidean graph on P with the property that all the edges incident to each point $p \in P$ lie in a wedge of angle α centred at p. For $\alpha = 2\pi/3$, the $\alpha - ST$ problem is NP-Hard. In the talk, we will discuss a 6-approximation and a 16/3-approximation algorithm.

In the last part, we will discuss approximation algorithms for the computation of maximum plane spanning trees in a bichromatic point set. A plane spanning tree in a geometric graph is a spanning tree that is non-crossing. Let R and B be two disjoint sets of points in the plane such that $R \cup B$ is in general position, and let $n = |R \cup B|$. A bichromatic plane spanning tree is a plane spanning tree in the complete bipartite geometric graph with bipartition (R, B). The maximum bichromatic plane spanning tree is a bichromatic plane spanning tree of the maximum total edge length. We will present an approximation algorithm with a ratio 1/4 that runs in $O(n \log n)$ time. We will also highlight what results are known for the computation of the maximum plane tree of a monochromatic point set and what we can say about the computation of the plane spanning tree of $R \cup B$ that minimizes the maximum degree.

*Joint work with Ahmad Biniaz, Prosenjit Bose, Jean-Lou De Carufel, Kimberley Crosbie, David Eppstein, Anna Lubiw, Patrick Morin, and Michiel Smid

†School of Computer Science, Carleton University, Ottawa, Canada, anil@scs.carleton.ca

.

Index of Authors