

Self-Approaching Graphs

Soroush Alamdari¹, Timothy M. Chan¹, Elyot Grant², Anna Lubiw¹, and
Vinayak Pathak¹

¹ Cheriton School of Computer Science, University of Waterloo, Waterloo, Canada
{s26hosse, tmchan, alubiw, vpathak}@uwaterloo.ca

² Massachusetts Institute of Technology, Cambridge, USA elyot@mit.edu

Abstract. In this paper we introduce *self-approaching* graph drawings. A straight-line drawing of a graph is *self-approaching* if, for any origin vertex s and any destination vertex t , there is an st -path in the graph such that, for any point q on the path, as a point p moves continuously along the path from the origin to q , the Euclidean distance from p to q is always decreasing. This is a more stringent condition than a greedy drawing (where only the distance between vertices on the path and the destination vertex must decrease), and guarantees that the drawing is a 5.33-spanner.

We study three topics: (1) recognizing self-approaching drawings; (2) constructing self-approaching drawings of a given graph; (3) constructing a self-approaching Steiner network connecting a given set of points.

We show that: (1) there are efficient algorithms to test if a polygonal path is self-approaching in \mathbb{R}^2 and \mathbb{R}^3 , but it is NP-hard to test if a given graph drawing in \mathbb{R}^3 has a self-approaching uv -path; (2) we can characterize the trees that have self-approaching drawings; (3) for any given set of terminal points in the plane, we can find a linear sized network that has a self-approaching path between any ordered pair of terminals.

Keywords: self-approaching, increasing-chord, graph drawing

1 Introduction

A straight-line graph drawing (or “geometric graph”) in the plane has points for vertices, and straight line segments for edges, where the weight of an edge is its Euclidean length. The drawing need not be planar. Rao *et al.* [27] introduced the idea of greedy drawings. A *greedy drawing* of a graph is a straight-line drawing in which, for each origin vertex s and destination vertex t , there is a neighbor of s that is closer to t than s is, i.e., there is a *greedy st -path* $P = (s = p_1, p_2, \dots, p_k = t)$ such that the Euclidean distances $D(p_i, t)$ decrease as i increases. This idea has attracted great interest in recent years (e.g. [1, 5, 16, 21, 24, 26]) mainly because a greedy drawing of a graph permits greedy local routing.

It is a very natural and desirable property that a path should always get closer to its destination, but there is more than one way to define this. Although every vertex along a greedy path gets closer to the destination, the same is not true of intermediate points along edges. See Figure 1.

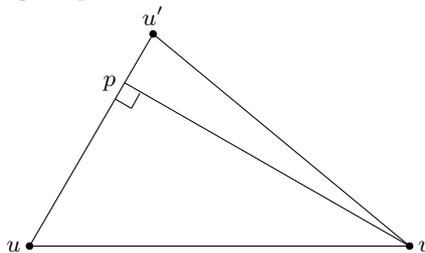


Fig. 1. As we move from u towards u' , distance to v first decreases (until p), then increases. However, $D(u', v) < D(u, v)$.

Another disadvantage of greedy paths is that the length of a greedy path is not bounded in terms of the Euclidean distance between the endpoints. This is another natural and desirable property for a path to have, and is captured by the *dilation* (or “stretch factor”) of a graph drawing—the maximum, over vertices s and t , of the ratio of their distance in the graph to their Euclidean distance. The dilation factor of greedy graph drawings can be unbounded.

Icking *et al.* [22] introduced a stronger notion of “getting closer” to a destination, that addresses both shortcomings of greedy paths. A curve from s to t is *self-approaching* if for any three points a, b, c appearing in that order along the curve, we have $D(a, c) \geq D(b, c)$. Icking *et al.* proved that a self-approaching curve has *detour* 5.3332, where the *detour* or *geometric dilation* of a curve is the supremum over points p and q on the curve, of the ratio of their distance along the curve to their Euclidean distance $D(p, q)$. This is stronger than dilation in that we consider all pairs of points, not just all pairs of vertices.

In this paper we introduce the notion of a *self-approaching* graph drawing—a straight-line drawing that contains, for every pair of vertices s and t , a self-approaching st -path and a self-approaching ts -path (which need not be the same). We also explore the related notion of an *increasing-chord* graph drawing, which has the stronger property that every pair of vertices is joined by a path that is self-approaching in both directions. Rote [28] proved that increasing-chord paths have dilation at most 4.77.

Our first result is a linear time algorithm to recognize a self-approaching polygonal path in the plane. We do not know the complexity of recognizing self-approaching graph drawings in the plane. However, we show that in \mathbb{R}^3 , it is NP-hard to test if a given graph drawing has a self-approaching uv -path, although we can still determine, in polynomial time, whether a single polygonal path is self-approaching.

Next, we consider the question of constructing a self-approaching drawing for a given graph. We give a linear time algorithm to recognize the trees that have self-approaching drawings. Finally, we consider the problem of connecting a given set of terminal points in the plane by a network that has a self-approaching path between every pair of terminals. We show that this can be done with a linear sized network.

2 Background

A *spanner* is a graph of bounded dilation. Spanners have been very well-studied—see for example the book by Narasimhan and Smid [25] and the survey by Eppstein [14]. A main goal is to efficiently construct a spanner on a given set of points, with the objective of minimizing dilation while keeping the number or total length of edges small. For recent results, see, e.g., [2, 15]. If Steiner vertices are allowed, their number should also be minimized, and different versions of the problem arise if we include the Steiner points in measuring the dilation, see [13].

The *detour* of a graph drawing is defined to be the supremum, over all points p, q of the drawing (whether at vertices, or interior to edges) of the ratio of their distance in the graph to their Euclidean distance. Note that if two edges cross in the drawing, then the detour is infinite. By contrast, a self-approaching drawing may have crossing edges, for example, any complete geometric graph is self-approaching. Constructing a network to minimize detour has also been considered [12, 11], though not as extensively as spanners.

Relevant background on greedy drawings is as follows. Answering a conjecture of Papadimitriou and Ratajczak [26], Leighton and Moitra [24] showed that any 3-connected planar graph has a greedy drawing, and Goodrich and Strash [16] reduced the number of bits needed for the coordinates in the embedding. Connecting the ideas of greedy drawings and spanners, Bose *et al.* [5] showed that every triangulation has an embedding in which local routing produces a path of bounded dilation.

Our problem of constructing a network with a self-approaching path between every pair of terminals is also related to Manhattan networks. Observe that an xy -monotone path is self-approaching. Thus, a network with an xy -monotone path between every pair of terminals is a self-approaching network. A *Manhattan network* has horizontal and vertical edges and includes an L_1 shortest path between every pair of terminals. So a Manhattan network is self-approaching. There is considerable work on finding Manhattan networks of minimum total length (so-called “minimum Manhattan networks”). There are efficient algorithms with approximation factor 2, and the problem has been shown to be NP-hard [10]. More relevant to us is the result of Gudmundsson *et al.* [18] that every point set admits a Manhattan network of $O(n \log n)$ vertices and edges, and there are point sets for which any Manhattan network has size at least $\Omega(n \log n)$.

For results on computing the dilation or detour of a path or graph, see the survey by Gudmundsson and Knauer [19], and the paper by Wulff-Nilsen [29].

The Delaunay triangulation has several good properties: it has dilation factor below 2 [30], and is a greedy drawing [6]. However, greedy paths in a Delaunay triangulation do not necessarily have bounded dilation. We conjecture that every Delaunay triangulation is self-approaching.

3 Preliminaries

We let $D(u, v)$ denote the Euclidean distance between points u and v in \mathbb{R}^d . A *curve* is a continuous function $f : [0, 1] \rightarrow \mathbb{R}^d$, and an *st-curve* is a curve f

with $f(0) = s$ and $f(1) = t$. An st -curve f is a *self-approaching curve* iff for all $0 \leq a \leq b \leq c \leq 1$ we have $D(f(a), f(c)) \geq D(f(b), f(c))$ (see Figure 2). Note that this definition is sensitive to the *direction* of the curve—an st -curve $f(x)$ may not be self-approaching even if the curve $f(1 - x)$ is.

An st -curve f has *increasing chords* iff for all $0 \leq a \leq b \leq c \leq d \leq 1$ we have $D(f(a), f(d)) \geq D(f(b), f(c))$, and we call f an *increasing-chord curve* if this is the case (see Figure 3 for an example). Note that $f(x)$ has increasing chords iff $f(1 - x)$ does, and that $f(x)$ and $f(1 - x)$ are both self-approaching whenever f has increasing chords. The converse also holds: a curve f has increasing chords whenever it is self-approaching in both directions, as we then have $D(f(a), f(d)) \geq D(f(a), f(c)) \geq D(f(b), f(c))$ for $0 \leq a \leq b \leq c \leq d \leq 1$.

Fig. 2. A self-approaching st -curve in \mathbb{R}^2 .

Fig. 3. An increasing-chord curve in \mathbb{R}^2 .

For convenience, we shall slightly abuse the above definitions for cases involving straight-line drawings of graphs. We shall say that a straight-line embedding (in \mathbb{R}^d) of a directed path $P = \{v_1, v_2, \dots, v_n\}$ from v_1 to v_n is self-approaching whenever it is the image of a self-approaching curve f with $f(0) = v_1$ and $f(1) = v_n$. Analogously, we say that the embedding of an undirected path has increasing chords whenever it is the image of an increasing-chord curve.

For the next series of definitions, let u and v be a pair of distinct points in \mathbb{R}^2 . We let \overline{uv} be the line passing through u and v , and we write l_{uv} to denote the line that passes through v and is perpendicular to \overline{uv} , noting that l_{uv} and l_{vu} are distinct parallel lines. We let l_{uv}^- denote the open half-plane containing u with boundary l_{uv} , and let l_{uv}^+ denote $\mathbb{R}^2 \setminus l_{uv}^-$, the closed half-plane that is the complement of l_{uv}^- (see Figure 4). Let $\text{slab}(uv)$ be the open strip bounded by l_{uv} and l_{vu} —in other words, the intersection of l_{uv}^- and l_{vu}^- .

The following characterization of self-approaching curves is straightforward:



Fig. 4. The line l_{uv} (dashed) and the area of l_{uv}^- (hatched).

Lemma 1. ([22]) *A piecewise-smooth curve f is self-approaching iff for each point $f(a)$, where $0 \leq a \leq 1$, the line perpendicular to f at $f(a)$ does not intersect any point $f(b)$ with $a < b \leq 1$.*

Corollary 1. *A piecewise-smooth curve f has increasing chords iff each line perpendicular to f at a point $f(a)$ intersects f only at $f(a)$.*

When dealing with straight-line drawings of graphs, we apply Lemma 1 to piecewise-linear curves. In this case, the lemma can be stated as follows:

Corollary 2. *Let $P = \{v_1, v_2, \dots, v_n\}$ be a directed path (directed from v_1 to v_n), embedded in \mathbb{R}^2 via straight line segments. Then, P is self-approaching iff for all $1 < i < j \leq n$, the point v_j lies in $l_{v_{i-1}v_i}^+$. Equivalently, P is self-approaching iff for all $1 < i \leq n$, the convex hull of $\{v_i, v_{i+1}, \dots, v_n\}$ lies in $l_{v_{i-1}v_i}^+$.*

Analogous characterizations are also possible in higher dimensions, with the half-planes $l_{v_{i-1}v_i}^+$ replaced by half-spaces bounded by hyperplanes orthogonal to $\overline{v_{i-1}v_i}$.

4 Testing whether paths are self-approaching

Corollary 2 implicitly suggests an algorithm to determine whether a directed path embedded in a Euclidean space is self-approaching. In this section, we provide improved algorithms for this task in two and three dimensions, as well as a lower bound. We assume a real RAM model in which all simple geometric operations can be performed in $O(1)$ time, and we assume that all straight-line drawings of paths $P = \{v_1, v_2, \dots, v_n\}$ are represented explicitly as lists of n points (requiring $O(n)$ space).

Theorem 1. *Given a straight-line drawing of a path $P = \{v_1, v_2, \dots, v_n\}$ in the plane, it is possible to test whether P is self-approaching in linear time.*

We prove this theorem by giving an $O(n)$ time algorithm to check that for all $1 < i \leq n$, the convex hull of v_i through v_n lies in $l_{v_{i-1}v_i}^+$. We can do this by iteratively processing and storing the convex hull. Details are provided in the appendix.

In three dimensions, we can obtain a similar result with slightly worse running time using an existing convex hull data structure that supports point insertion and half-space range emptiness queries. Again, the proof is deferred to the appendix.

Theorem 2. *Given a straight-line drawing of a path $P = \{v_1, v_2, \dots, v_n\}$ in \mathbb{R}^3 , it is possible to test whether P is self-approaching in $O(n \log^2 n / \log \log n)$ time.*

Next, we show that Theorem 2 is tight up to a factor of $\log n / \log \log n$ by proving a lower bound of $\Omega(n \log n)$ on the running time of any algorithm for determining whether a directed path embedded in \mathbb{R}^3 is self-approaching. We do this by reducing from the *set intersection problem*, for which a solution requires $\Omega(n \log n)$ time on an input of size n in the algebraic computation tree model [3]. We can show the following:

Theorem 3. *Given a straight-line drawing of a path $P = \{v_1, v_2, \dots, v_n\}$ in \mathbb{R}^3 , at least $\Omega(n \log n)$ time is required to test whether P is self-approaching in the algebraic computation tree model.*

To prove this, we build an embedding of a path in \mathbb{R}^3 using ‘cannons’ and ‘targets’, where a slab perpendicular to a ‘cannon’ collides with a ‘target’ if and only if the corresponding elements of the sets A and B are identical. Details are provided in the appendix.

The same construction also yields the following:

Corollary 3. *Given a straight-line drawing of a path $P = \{v_1, v_2, \dots, v_n\}$ in \mathbb{R}^3 , at least $\Omega(n \log n)$ time is required to test whether P has increasing chords in the algebraic computation tree model.*

5 Finding self-approaching paths in graphs

We do not know whether it is possible to determine, in polynomial time, whether a given source and sink in a planar straight-line drawing can be connected via a self-approaching path. However, by employing the cannons and targets introduced in Section 4, we can show that the problem is hard in three or more dimensions:

Theorem 4. *Given a straight-line drawing of a graph G in \mathbb{R}^3 , and a pair of vertices s and t from G , it is NP-hard to determine if a self-approaching st -path exists. It is also NP-hard to determine if an increasing-chord st -path exists.*

To prove this theorem, we reduce from 3SAT. Our proof uses similar ‘cannons’ and ‘targets’ to those used in the proof of Theorem 3, but this time, the cannons correspond to variable assignments and the targets correspond to literals in clauses. The full proof is found in the appendix.

6 Recognizing graphs having self-approaching drawings

In this section we examine self-approaching drawings of trees. The following characterization is quite natural:

Lemma 2. *In a self-approaching drawing of a tree T , for each edge (u, v) , there is no edge or vertex of $T \setminus uv$ that intersects $\text{slab}(uv)$.*

Proof. Since there is a unique path connecting vertices s and t in any tree T , a drawing of T is self-approaching if and only if it has increasing chords. The result then follows from Corollary 1. \square

As it turns out, we can quickly determine whether a tree admits a self-approaching drawing:

Theorem 5. *Given a tree T , in linear time we can decide whether or not T admits a self-approaching drawing.*

Proof. To prove this theorem, we completely characterize trees that admit self-approaching drawings. We require two definitions of special graphs.

A *windmill* having *sweep length* k is a tree constructed by subdividing the edges of $K_{1,3}$ $k - 1$ times iteratively and then attaching a leaf to each subdivision vertex. The term *sweep* shall denote one of the three new disjoint paths of length k that the replace edges of $K_{1,3}$ during the subdivision process. A windmill is depicted in Figure 5.

Fig. 5. A windmill with sweeps of length 2.

The *crab graph* is the tree with vertices $\{a, b, a_1, a_2, b_1, b_2, a_{11}, a_{12}, a_{21}, a_{22}, b_{11}, b_{12}, b_{21}, b_{22}\}$ and edges $\{(a, b), (a, a_1), (a, a_2), (b, b_1), (b, b_2), (a_1, a_{11}), (a_1, a_{12}), (a_2, a_{21}), (a_2, a_{22}), (b_1, b_{11}), (b_1, b_{12}), (b_2, b_{21}), (b_2, b_{22})\}$ as depicted in Figure 6. A graph G is crab-free only if it has no subgraph that is isomorphic to some subdivision of a crab.

We prove Theorem 5 in two steps. Write Δ_T for the maximum degree of a vertex in T .

1. First we show that T with $\Delta_T \geq 4$ admits a self-approaching drawing if and only if T is a subdivision of $K_{1,4}$.
2. Then we show that T with $\Delta_T \leq 3$ admits a self-approaching drawing if and only if it is a subdivision of a windmill, which happens if and only if T is crab-free.

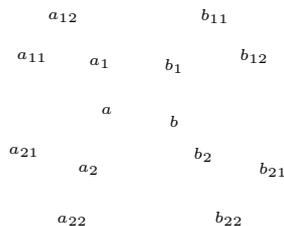


Fig. 6. An embedding of the crab.

To establish the first result, we prove the following in the appendix:

Lemma 3. *In an increasing-chord drawing of a path, the sum of the sizes of the angles in any consecutive chain of k left turns (or right turns) is at least $\pi(k-1)$ if $k > 1$ and at least $\pi/2$ if $k = 1$.*

Corollary 4. *If T admits a self-approaching drawing, then $\Delta_T \leq 4$. Also, if $\Delta_T = 4$, then there is only one vertex of degree 4 in T , and the four angles at the vertex of degree 4 have all size $\pi/2$, and the rest of the angles have size π .*

This concludes the first step of the proof. For the second step, we prove the following three structural lemmas, which establish the equivalence of a tree being a subdivision of a windmill, being crab-free, and admitting a self-approaching drawing. Proofs are located in the appendix.

Lemma 4. *Let T be a tree with $\Delta_T \leq 3$ that is crab-free. Then T is a subdivision of a windmill.*

Lemma 5. *Let T be a tree that is a subdivision of a windmill. Then T admits a self-approaching drawing.*

Lemma 6. *Let T be a tree that contains a subdivision of the crab. Then T does not admit a self-approaching drawing.*

Combining these results, we obtain the second step of the proof of the theorem. This completes the characterization of all trees that admit self-approaching drawings. To complete the proof of the Theorem 5, it suffices to observe that it is possible, in linear time, to check whether a tree T is a subdivision of $K_{1,4}$ or is a subdivision of a windmill.

7 Constructing self-approaching Steiner networks

We now turn our attention to the following problem: Given a set P of points in a plane, draw a graph N with straight edges and $P \subseteq V(N)$ such that for each ordered pair of points $p, q \in P$ there is a self-approaching path from p to q in the drawing of N . We call the points in $V(N) \setminus P$ *Steiner points* and the graph N a *self-approaching Steiner network*. An increasing-chord Steiner network is defined similarly.

We show that small increasing-chord Steiner networks (and hence small self-approaching Steiner networks) can always be constructed for any given set of points in the plane.

Theorem 6. *Given a set P of n points in the plane, there exists an increasing-chord Steiner network having $O(n)$ vertices and edges.*

Proof. Given points p and q , let θ_{pq} denote the angle between the line pq and the x -axis (we take the smaller of the two angles formed, so that $\theta_{pq} \in [0, \pi/2]$). A path is *xy-monotone* if every vertical line intersects the path at most once and every horizontal line intersects the path at most once. Clearly, an *xy-monotone* path is self-approaching. We will construct a linear-size Steiner network G with the following property:

For every pair of points $p, q \in P$ with $\theta_{pq} \in [\pi/8, 3\pi/8]$, there is an *xy-monotone* path from p to q in G .

To handle the remaining pairs of points, we can rotate the coordinate axes by $\pi/4$ and apply the same construction to obtain another Steiner network G' . We can then return the union of G and G' .

To construct G , we first build a *quadtrees* [20], defined as follows: The root stores an initial square enclosing P . At each node, we divide its square into four congruent subsquares and create a child for each subsquare that is not empty of points of P . The tree has n leaves.

To ensure that the tree has $O(n)$ internal nodes, we compress each maximal path of degree-1 nodes by keeping only the first and last node in the path. The result is a *compressed quadtree*, denoted T .

For each square B in the compressed quadtree T , we add the four corner vertices and edges of B to G . (Note that we allow overlapping edges in our construction; it is not difficult to avoid overlaps by subdividing the edges appropriately.) For each leaf square B in T containing a single point $p \in P$, we add a 2-link *xy-monotone* path in G from p to each corner of B . For each degree-1 square B in T having a single child square B' , we add a 2-link *xy-monotone* path in G from each corner of B' to the corresponding corner of B . By induction, it then follows that for every point $p \in P$ inside a square B in T , there is an *xy-monotone* path in G from p to each corner of B . The number of vertices and edges in G thus far is $O(n)$.

Given a parameter $\varepsilon > 0$, a *well-separated pair decomposition* of P is a collection of pairs of sets $\{A_1, B_1\}, \dots, \{A_s, B_s\}$, such that³

1. for every pair of points $p, q \in P$, there is a unique index i with $(p, q) \in A_i \times B_i$ or $(p, q) \in B_i \times A_i$;
2. A_i and B_i are *well-separated* in the sense that both the diameter of A_i and the diameter of B_i is at most $\varepsilon d(A_i, B_i)$, where $d(A_i, B_i)$ is the minimum distance between A_i and B_i .

³ In the original definition [7], A_i and B_i are subsets of P , but for our purposes, we will take A_i and B_i to be regions in the plane (namely, squares).

It is known that a well-separated pair decomposition consisting of $s = O(n/\varepsilon^2)$ pairs always exists [7]. Furthermore, such a decomposition can be constructed by a simple quadtree-based algorithm (for example, see [20] or [8]), where the sets A_i and B_i are in fact squares appearing in the compressed quadtree T .

To finish the construction of G , we consider each pair $\{A_i, B_i\}$ in the decomposition such that A_i and B_i are separated by both a vertical line and a horizontal line. Without loss of generality, suppose that A_i is to the left of and below B_i . We add a 2-link xy -monotone path in G from the upper right corner of A_i to the lower left corner of B_i . The overall number of vertices and edges in G is $O(n/\varepsilon^2)$.

To show that G satisfies the stated property, let $p, q \in P$ with $\theta_{pq} \in [\pi/8, 3\pi/8]$. Suppose that $(p, q) \in A_i \times B_i$. If A_i and B_i are intersected by a common horizontal line, then θ_{pq} must be upper-bounded by $O(\varepsilon)$ because A_i and B_i are well-separated; this is a contradiction if we make the constant ε sufficiently small. Thus, A_i and B_i must be separated by a horizontal line, and similarly by a vertical line via a symmetric argument. Without loss of generality, suppose that A_i is to the left of and below B_i . By concatenating xy -monotone paths in G , we can get from p to the upper right corner of A_i , then to the lower left corner of B_i , and finally to q . \square

In the above construction, the edges we add for each well-separated pair $\{A_i, B_i\}$ may cross other edges, although it is possible to modify the construction to ensure that the network G is planar (and similarly G'). However, we do not know how to avoid crossings in the final network obtained by unioning G and G' , while keeping the number of edges linear.

Whether planar self-approaching Steiner networks of linear size can be constructed or not is an interesting question. We consider the Delaunay triangulation to be a potential candidate, and conjecture the following:

Conjecture 1. The Delaunay triangulation of T is a self-approaching drawing with zero Steiner points.

Unfortunately, the conjecture does not hold for increasing-chord networks (a counterexample is presented in the appendix):

Theorem 7. *There exists a set of six points in the plane whose Delaunay triangulation is not an increasing-chord drawing.*

8 Conclusions

We have introduced the notion of self-approaching and increasing-chord graph drawings, with rich connections to greedy drawings, spanners, dilation and detour, and minimum Manhattan networks.

Our results are preliminary. We leave open the following questions:

- Can we test, in polynomial time, if a straight-line graph drawing in the plane is self-approaching [or increasing-chord]? Or is the problem NP-complete?

- Given a graph G , can we efficiently produce a self-approaching drawing of G if one exists? Can we produce a drawing in which a self-approaching path between any pair of vertices can be found via local routing?
- Does every 3-connected planar graph have a self-approaching [or increasing-chord] drawing? We have some hope that Goodrich and Strash’s greedy drawing method for 3-connected planar graphs [16] in fact produces an increasing-chord drawing, and furthermore, that their local routing strategy produces increasing-chord paths. This would have the significant implication that every 3-connected planar graph has an embedding where local routing gives paths of bounded detour (hence bounded dilation). Bose *et al.* [5] recently proved the weaker result that every triangulation has an embedding where local routing gives paths of bounded dilation.
- Is the Delaunay triangulation self-approaching?

Acknowledgements. Anna Lubiw thanks Marcus Brazil, Victor Chepoi, and Martin Zachariasen for workshop discussions that inspired this line of enquiry. This work was done as part of an Algorithms Problem Session at the University of Waterloo, and we thank the other participants for helpful discussions.

References

1. P. Angelini, F. Frati, and L. Grilli. An algorithm to construct greedy drawings of triangulations. *J. Graph Algorithms Appl.*, 14(1):19–51, 2010.
2. B. Aronov, M. de Berg, O. Cheong, J. Gudmundsson, H. Haverkort, M. Smid, and A. Vigneron. Sparse geometric graphs with small dilation. *Computational Geometry*, 40(3):207 – 219, 2008.
3. M. Ben-Or. Lower bounds for algebraic computation trees. In *Proc. 15th ACM Symposium on Theory of Computing*, pages 80–86, New York, 1983.
4. J. L. Bentley and J. B. Saxe. Decomposable searching problems I: Static-to-dynamic transformations. *J. Algorithms*, 1:301–358, 1980.
5. P. Bose, R. Fagerberg, A. van Renssen, and S. Verdonschot. Competitive routing in the half- θ_6 -graph. In *Proc. 23rd ACM-SIAM Symposium on Discrete Algorithms*, pages 1319–1328, 2012.
6. P. Bose and P. Morin. Online routing in triangulations. *SIAM J. Comput.*, 33(4):937–951, 2004.
7. 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*, 42:67–90, 1995.
8. T. M. Chan. Well-separated pair decomposition in linear time? *Inform. Process. Lett.*, 107:138–141, 2008.
9. B. Chazelle. An optimal algorithm for intersecting three-dimensional convex polyhedra. *SIAM J. Comput.*, 21(4):671–696, 1992.
10. F. Y. L. Chin, Z. Guo, and H. Sun. Minimum Manhattan network is NP-complete. *Discrete & Computational Geometry*, 45(4):701–722, 2011.
11. A. Dumitrescu and C. D. Tóth. Light orthogonal networks with constant geometric dilation. *Journal of Discrete Algorithms*, 7(1):112–129, 2009.
12. A. Ebbels-Baumann, A. Grune, and R. Klein. The geometric dilation of finite point sets. *Algorithmica*, 44:137–149, 2006.

13. A. Ebbers-Baumann, A. Grüne, R. Klein, M. Karpinski, C. Knauer, and A. Lingas. Embedding point sets into plane graphs of small dilation. *Int. J. Comput. Geometry Appl.*, 17(3):201–230, 2007.
14. D. Eppstein. Spanning trees and spanners. In J. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 425–461. North-Holland, 2000.
15. P. Giannopoulos, R. Klein, C. Knauer, M. Kutz, and D. Marx. Computing geometric minimum-dilation graphs is NP-hard. *Int. J. Comput. Geometry Appl.*, 20(2):147–173, 2010.
16. M. T. Goodrich and D. Strash. Succinct greedy geometric routing in the Euclidean plane. In *Proc. 20th International Symposium on Algorithms and Computation*, pages 781–791, 2009.
17. R. L. Graham. An efficient algorithm for determining the convex hull of a finite planar set. *Inform. Process. Lett.*, 1:132–133, 1972.
18. J. Gudmundsson, O. Klein, C. Knauer, and M. Smid. Small Manhattan networks and algorithmic applications for the earth mover’s distance. In *Proc. 23rd European Workshop on Computational Geometry*, pages 174–177, 2007.
19. J. Gudmundsson and C. Knauer. Dilation and detour in geometric networks. In T. Gonzalez, editor, *Handbook on Approximation Algorithms and Metaheuristics*. Chapman & Hall/CRC Press, 2007.
20. S. Har-Peled. *Geometric Approximation Algorithms*. AMS, 2011.
21. X. He and H. Zhang. On succinct convex greedy drawing of 3-connected plane graphs. In *Proc. 22nd ACM–SIAM Symposium on Discrete Algorithms*, pages 1477–1486, 2011.
22. C. Icking, R. Klein, and E. Langetepe. Self-approaching curves. *Math. Proc. Camb. Phil. Soc.*, 125:441–453, 1995.
23. D. G. Kirkpatrick. Optimal search in planar subdivisions. *SIAM J. Comput.*, 12(1):28–35, 1983.
24. T. Leighton and A. Moitra. Some results on greedy embeddings in metric spaces. *Discrete and Computational Geometry*, 44:686–705, 2010.
25. G. Narasimhan and M. Smid. *Geometric Spanner Networks*. Cambridge University Press, 2007.
26. C. H. Papadimitriou and D. Ratajczak. On a conjecture related to geometric routing. *Theor. Comput. Sci.*, 344:3–14, 2005.
27. A. Rao, S. Ratnasamy, C. Papadimitriou, S. Shenker, and I. Stoica. Geographic routing without location information. In *Proc. 9th International Conference on Mobile Computing and Networking*, pages 96–108, 2003.
28. G. Rote. Curves with increasing chords. *Mathematical Proceedings of the Cambridge Philosophical Society*, 115:1–12, 1994.
29. C. Wulff-Nilsen. Computing the maximum detour of a plane geometric graph in subquadratic time. *Journal of Computational Geometry*, 1(1):101–122, 2010.
30. G. Xia. Improved upper bound on the stretch factor of Delaunay triangulations. In *Proc. 27th ACM Symposium on Computational Geometry*, pages 264–273, 2011.
31. A. C. Yao. Lower bounds for algebraic computation trees with integer inputs. *SIAM J. Comput.*, 20(4):655–668, 1991.

A Appendix

A.1 Proof of Theorem 1

Theorem 1 Given a straight-line drawing of a path $P = \{v_1, v_2, \dots, v_n\}$ in the plane, it is possible to test whether P is self-approaching in linear time.

Proof. By Corollary 2, we must only check that for all $1 < i \leq n$, the convex hull of v_i through v_n lies in $l_{v_{i-1}v_i}^+$. We can do all of these checks in $O(n)$ time by performing them iteratively, beginning with $i = n$ and processing the points in decreasing order. While doing this, we will either show that P is not self-approaching, or we will be able to use the properties of self-approaching paths to construct the convex hull of the traversed vertices incrementally in linear total time by an algorithm similar to Graham's scan [17].

We now describe a step of the algorithm. Assume that the directed path $P_i = \{v_i, \dots, v_n\}$ is self-approaching and assume the convex hull C of vertices v_i through v_n has already been computed and is stored by keeping track of the neighbors of each vertex on its boundary. Since P_i is self-approaching, point v_i must lie on the boundary of C . Let v_i^1 and v_i^2 be the neighbors of v_i in C . We can check to see if $l_{v_{i-1}v_i}$ intersects C in $O(1)$ time, because it is only necessary to check if line segments $\overline{v_i v_i^1}$ and $\overline{v_i v_i^2}$ intersect $\text{slab}(v_{i-1}v_i)$. If an intersection is found, then P is not self-approaching and we can terminate the algorithm. Otherwise, we add v_{i-1} to C and recompute the convex hull. This can be done by repeatedly removing the vertices of C on both sides of v_i until convex angles are obtained, after which the convex hull of $C \cup \{v_{i-1}\}$ will remain. Each vertex in P will be removed at most once from a convex hull in some step of the algorithm, so the total running time for all steps of the algorithm is $O(n)$. \square

A.2 Proof of Theorem 2

Theorem 2 Given a straight-line drawing of a path $P = \{v_1, v_2, \dots, v_n\}$ in \mathbb{R}^3 , it is possible to test whether P is self-approaching in $O(n \log^2 n / \log \log n)$ time.

Proof. The proof is analogous to that of Theorem 1, with the only change being that we must employ a more complicated data structure to store the convex hull and test whether it intersects a given half-space range. For each edge $v_{i-1}v_i$, we can ensure that $\text{slab}(v_{i-1}v_i)$ does not intersect the convex hull C by performing two half-space range emptiness queries on C . If no intersection is found, then we may insert point v_{i-1} to our data structure and perform the next iteration of the algorithm. If the algorithm successfully inserts all points into C , then the path P must be self-approaching.

Achieving the stated running time requires a nontrivial data structure combining several known ideas. There is a static data structure for half-space range emptiness in \mathbb{R}^3 with $O(n)$ space and $O(\log n)$ query time, by reduction to *planar point location* in dual space [23]; the preprocessing time is $O(n)$ if we are given the convex hull. The static data structure can be transformed into a semidynamic

data structure with $O(b \log_b n)$ amortized insertion time and $O(\log_b n \log n)$ query time for a given parameter b , by known techniques—namely, a b -ary version of Bentley and Saxe’s *logarithmic method* [4], using Chazelle’s linear-time algorithm for merging two convex hulls [9] as a subroutine. By setting $b = \log n$, both amortized insertion time and query time are bounded by $O(\log^2 n / \log \log n)$, yielding the desired result. \square

A.3 Proof of Theorem 3

Theorem 3 Given a straight-line drawing of a path $P = \{v_1, v_2, \dots, v_n\}$ in \mathbb{R}^3 , at least $\Omega(n \log n)$ time is required to test whether P is self-approaching in the algebraic computation tree model.

Proof. We first need a few gadgets for our reduction. Let $\beta = \pi/6$ and $\alpha = 1$. For a point $p \in \mathbb{R}^2$, we define a *cannon* c at p to be an embedding of a 3-vertex path $[c^0, c^1, c^2]$ where the points are located as follows:

- c^0 is placed at p ,
- c^2 is placed at $p + (1, 0)$, that is, α units to the right of p , and
- c^1 is placed at $p + (3/4, \sqrt{3}/4)$, on the line that meets the x -axis at an angle β and passes through c^0 , such that the angle $\widehat{c^0 c^1 c^2}$ is a right angle.

Similar to a cannon, a *target* t at point p with respect to the a ℓ is an embedding of a 3-vertex path $[t^0, t^1, t^2]$, where the points in t are positioned as follows:

- t^0 is placed at p ,
- t^1 at the intersection of ℓ and ℓ' , where ℓ' is the line of slope 1 passing through t^0 , and
- t^2 is placed on the x -axis such that the angle $\widehat{t^0 t^1 t^2}$ is a right angle.

With these gadgets in hand, we now present a reduction from the set intersection problem. Let \mathcal{I} be an instance of the set intersection problem, where we are asked to check if there is a common element in sets A and B . Using Yao’s improvement to Ben-Or’s lower bound constructions for algebraic computation trees [31], it suffices to consider the case where A and B are sets of non-negative integers. Letting M be the maximum element in A and B , we first divide each element of A and B by $2M/\pi$ so that both A and B are subsets of $[0, \pi/2]$, noting that this can be done in linear time. Let $\varepsilon < \pi/2M$ so that $|a - b| > \varepsilon$ for all $a, b \in A \cup B$ with $a \neq b$, and let γ be a sufficiently large constant (depending on M). Using the coordinates of A and B , we embed a path $P = \{v_0, v_1, v_2, \dots, v_{1+2|A|+2|B|}\}$ in \mathbb{R}^3 as follows:

1. Start with the vertex v_0 placed on the origin.
2. For each $1 \leq i \leq |A|$, place a cannon c_i in the xy -plane, attached to the current path, with $c_1^0 = v_0$ and $c_i^0 = c_{i-1}^2$ for $i \geq 2$. Cannon c_i shall represent the element $a_i \in A$. At this stage, the path should appear as a chain of $|A|$ cannons lined up along the x -axis.

3. Place the next vertex $v_{2|A|+1}$ of the path on $(\alpha|A| + \gamma, 0)$.
4. For each $1 \leq i \leq |B|$, add a target t_i in the xy -plane, placed at the end of the current path with respect to $\ell = \overline{v_0v_1}$. Target t_i shall represent the element $b_i \in B$ and the targets, like the cannons, shall be aligned along the x -axis (Figure 7 shows what the path looks like at this point).
5. Modify the embedding by rotating each cannon about the x -axis through an angle a_i (in other words, relocate p_i^1 from $(x, 3/4, 0)$ to $(x, 3/4 \cos(a_i), 3/4 \sin(a_i))$).
6. Similarly, rotate each target t_i^1 about the x -axis through an angle b_i by relocating t_i^1 .
7. Let P be the path obtained after these rotations.

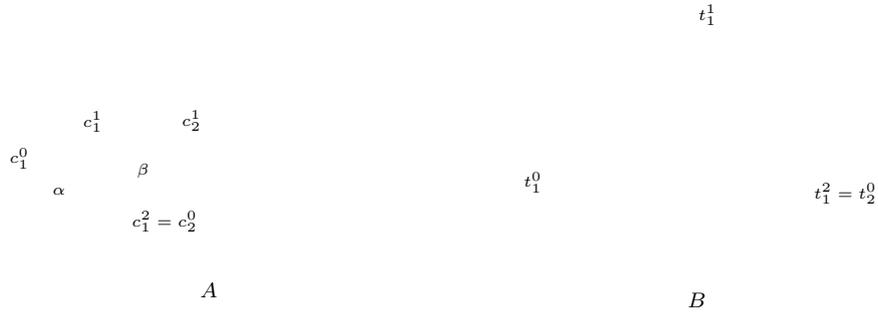


Fig. 7. The cannons (left) and the targets (right).

Fig. 8. Placement of a target.

The key idea of our proof is that P shall be a self-approaching path (in the v_0 to $v_{1+2|A|+2|B|}$ direction) if and only if A and B do not intersect. More

specifically, $\text{slab}(c_i^1 c_i^2)$ shall collide with the target t_j if and only if element a_i equals element b_j .

Only if: Assume $a_i = b_j$. It is then easy to see that $\text{slab}(c_i^1 c_i^2)$ collides with the target t_j , since both the cannon c_i and the target t_j are rotated around the x -axis through the same angle. It follows, by Lemma 1, that P is not self-approaching.

If: By Lemma 1, it suffices to show that if A and B do not intersect, then for any edge e in P , $\text{slab}(e)$ does not intersect any edges in the path after e . It is straightforward from our construction that the only way such an intersection can occur is if $\text{slab}(c_i^1 c_i^2)$ intersects a point t_j^1 for some i and j . Let s be $\text{slab}(c_{|A|}^1 c_{|A|}^2)$ as it is positioned prior to step 5 in the construction. Define θ to be the minimum amount that we need to rotate the target t_1 , so that the point t_1^1 does not lie in s . It is easy to see that θ decreases as γ increases, and more specifically that $\lim_{\gamma \rightarrow \infty} \theta = 0$. Therefore, we can choose γ large enough (with respect to ε), so that $\text{slab}(c_i^1 c_i^2)$ intersects t_j if and only if $|a_i - b_j| < \varepsilon$, which, by construction, happens only when $a_i = b_j$. The result follows. \square

A.4 Proof of Theorem 4

Theorem 4 Given a straight-line drawing of a graph G in \mathbb{R}^3 , and a pair of vertices s and t from G , it is NP-hard to determine if a self-approaching st -path exists. It is also NP-hard to determine if an increasing-chord st -path exists.

Proof. We establish the result for the case of self-approaching paths; the proof for the increasing-chord case is similar. We reduce from 3SAT. Let \mathcal{I} be an instance of 3SAT. Let $\{x_1, x_2, \dots, x_n\}$ be the variables in \mathcal{I} . For any $1 \leq k \leq n$, let the literal y_k be the negation of the literal z_k , both associated with the boolean variable x_k . Let $\{w_1, w_2, \dots, w_m\}$ be the set of clauses associated with \mathcal{I} , where $w_i = \{w_i^1, w_i^2, w_i^3\}$ and each literal w_i^j is either y_k or z_k for some value of k . Let $\varepsilon = \pi/2n$. We draw the graph G as follows:

1. Place the vertex s at the origin.
2. Place two cannons c_1 and c_2 corresponding to y_1 and z_1 , both at s .
3. For all $1 < i \leq n$, place two cannons c_{2i-1} and c_{2i} corresponding to y_i and z_i , both at the point $c_{2i-2}^2 = c_{2i-3}^2$.
4. Place a vertex s' at $(\alpha n + \gamma, 0)$, adjacent to c_{2n}^2 .
5. Place three targets t_1, t_2 and t_3 at s' with respect to the line $\overline{sc_1^1}$.
6. For all $1 \leq i \leq m$, place three targets t_{3i-2}, t_{3i-1} and t_{3i} at t_{3i-3}^2 , with respect to the line $\overline{sc_1^1}$.
7. For all $1 \leq i \leq 2n$, rotate c_i^1 about x -axis through an angle of $i\varepsilon$.
8. For all $1 \leq i \leq m$ and $1 \leq j \leq 3$, suppose that $w_i^j = y_k$ (respectively, z_k). Then rotate $t_{3(i-1)+j}^1$ about the x -axis through an angle of $(2k-1)\varepsilon$ (respectively, $2k\varepsilon$)—in other words, rotate $t_{3(i-1)+j}^1$ through the same amount that the cannon corresponding to the value of the literal w_i^j is rotated, so that a cannon ‘hits’ a target if and only if the cannon and target correspond to the same literal.

The rest of the proof is quite similar to the proof of Lemma 3. In particular, we shall show that \mathcal{I} is satisfiable if and only if there is a self-approaching path from s to t_{3m}^2 . We will reuse the following statement from the proof of Lemma 3: for $1 \leq i \leq n$, $\text{slab}(c_i^1 c_i^2)$ intersects the target t_j , if and only if t_j^1 and c_i^1 are rotated by the same amount, hence correspond to the same literal. Let P be a path from s to t_{3m}^2 . Assume P is a self-approaching path. For each cannon c_i appearing in P , assign the literal corresponding to c_i to be false, and its negation to be true. Then, it is easy to show that in each clause, there is at least one true literal: the one appearing in P . Similar to this, from a satisfying assignment of the variables, we can construct a self-approaching path by taking the cannons corresponding to false literals. For the second part of the path, we use one of the three targets assigned to each clause: one that corresponds to a true literal. This way, since each target that is traversed in P corresponds to a cannon that was not traversed in P , P would be a self-approaching path.

The same proof also works to establish NP-hardness for finding an increasing chord st -path. Note that this is because the drawing of the graph is constructed in a way that any increasing-chord path connecting s to t_{3m}^2 is a self-approaching path in the s -to- t_{3m}^2 direction and vice versa. \square

A.5 Proof of Lemma 3

Lemma 3 In an increasing-chord drawing of a path, the sum of the sizes of the angles in any consecutive chain of k left turns (or right turns) is at least $\pi(k-1)$ if $k > 1$ and at least $\pi/2$ if $k = 1$.

Proof. There is clearly no angle smaller than $\pi/2$ in any increasing-chord drawing of a path. Let (u', u) and (v, v') be the first and last edges of the chain. Let s be the point in the plain such that $\widehat{uu's}$ and $\widehat{vv's}$ are right angles (See Figure 9). \widehat{usv} is greater than π or else either $l_{uu'}$ or $l_{vv'}$ intersects uv -path. Since the sum of the angles of a simple polygon of n vertices is $\pi(n-1)$, the sum of the angles of vertices along the uv -path is greater than $\pi(k-2)$ if the length of the path is k . \square

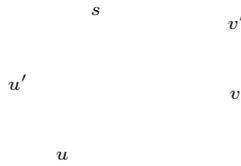


Fig. 9. For proof of Lemma 3.

A.6 Proof of Lemma 4

Lemma 4 Let T be crab-free tree with $\Delta_T \leq 3$. Then T is a subdivision of a windmill.

Proof. We assume there is a vertex s that has three disjoint paths to three degree-3 vertices. Otherwise the graph is a subgraph of a subdivision of a sweep. Note that vertices a and b of the crab graph satisfy this property. There is only a single such vertex s in a tree that admits a self-approaching drawing, otherwise the graph would have a subdivision of the crab graph as a subgraph. Put s on the center of the windmill. In each component of the graph obtained by removing s , there is a path going through all degree-3 vertices (otherwise, there would be another degree-3 vertex that satisfies the property). Put degree three vertices on sweeps such that if u is on the path from v to w in G , u is still between v and w on the sweep. Then add degree two and degree one vertices on the edges. \square

A.7 Proof of Lemma 5

Lemma 5 Let T be a tree that is a subdivision of a windmill. Then T admits a self-approaching drawing.

Proof. It suffices to show that any windmill admits a self-approaching drawing. We draw a $K_{1,3}$ so that each angle is $2\pi/3$ and edges are equal in length. From each leaf l , draw two rays so that the angle between them is $\pi/2 + \varepsilon\pi$ for some small ε and each of the angles formed by a ray and the leaf is $3\pi/2 - \varepsilon\pi/2$. It can be easily seen that for each of the two rays at l , the half-plane consisting of the lines perpendicular to that ray does not intersect any of the other two leaves of $K_{1,3}$.



Fig. 10. Embedding a sweep of length 3. The two rays are drawn using dashed segments and α here is $\varepsilon\pi/2t$.

Let γ be a number to be set later. For each leaf l of the drawing of $K_{1,3}$, we draw the sweep of l as follows: Assume that l has a sweep of length t . We draw the sweep that contains l inside the two rays extending from l . We do so in a way that each edge, when translated to have one of its endpoints on l , lies

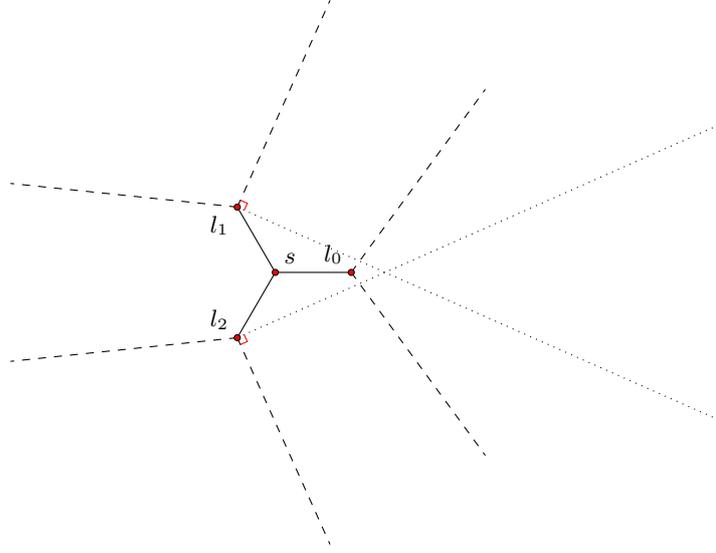


Fig. 11. The drawing of s and its three neighbors (solid lines) along with the two rays of the neighbors (dashed) for $\varepsilon = 0.1$. The constructed drawing should be such that the sweep attached to l_0 lies completely inside the area next to l_0 that is bounded by dashed and dotted lines.

entirely inside the area bounded by the two rays. This means if the sweep is small enough, we can avoid intersection with any strip l_e of some edge e belonging to another sweep. To do so, draw the first edge of the sweep so that it has length γ and makes a $\varepsilon\pi/2$ angle with one of the rays at l . Continue to draw the whole sweep with each edge having a $\frac{\varepsilon\pi}{2t}$ difference of direction with the previous edge and length γ (See Figure 10). This means that the last edge of the sweep is parallel to one of the two rays at l .

To make sure that the strip parallel to each edge is empty, γ can be set to $\sin(\pi/4 - \varepsilon\pi/2 - \pi/6)/t$.

Draw the leaf attached to l so that it is inside the reflex angle at l and lies exactly on one of the rays. Then draw rest of the leaves attached to the vertices of the sweep, in a way that each new edge is exactly in the middle of the reflex angle of the two edges of the sweep (See Figure 12). The length of each of these new edges should be small enough so that none of them is inside the strip induced by another one. To satisfy this, the length of each such leaf can be $\gamma \sin(\varepsilon\pi/4t)$. \square

A.8 Proof of Lemma 6

Lemma 6 Let T be a tree that contains a subdivision of the crab. Then T does not admit a self-approaching drawing.

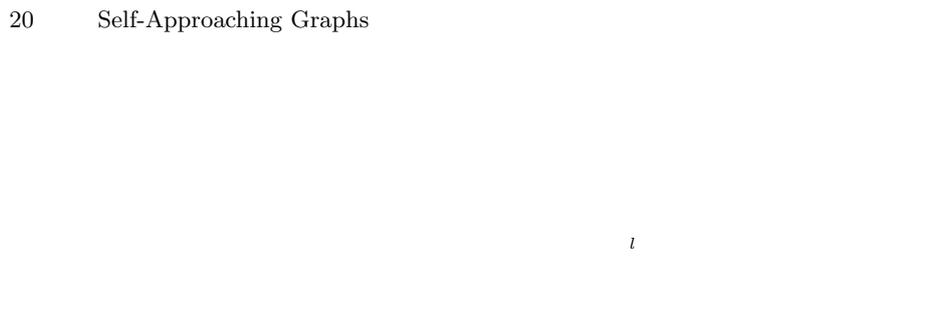


Fig. 12. Embedding the leaves of a sweep. The two rays are drawn using dashed segments.

Proof. It is easy to see that if a tree admits a self-approaching drawing, then any connected subgraph of it also admits a self-approaching drawing. Therefore, we only need to show that no subdivision of the crab graph has a self-approaching drawing. For contradiction, suppose that the crab graph that has a self-approaching drawing. By choice of C , there are no vertices of degree 2 adjacent to a leaf of C . By Lemma 3, the total size of the chain of four angles on path from $a_{1,2}$ to $b_{1,1}$ is greater than 3π . By similar arguments, the angles on the path from $a_{2,2}$ to $b_{2,2}$ also sum to 3π . Similarly, by Lemma 3, the total size of the chain of three consecutive angles on the path from $a_{1,1}$ to $a_{2,1}$ is greater than 2π . By similar arguments, the size of the angles on the path from $b_{1,2}$ to $b_{2,1}$ also sum to 2π . By Lemma 3, each of the four angles formed by the eight leaves has size at least $\pi/2$, summing to 2π . This adds up to a total strictly greater than $3\pi + 3\pi + 2\pi + 2\pi + 2\pi = 12\pi$. Since these angles are the angles around the 6 vertices a, b, a_1, a_2, b_1 , and b_2 , we have a contradiction.

Now consider C to be the smallest subdivision of the crab graph that admits a self-approaching drawing. By choice of C , there are no vertices of degree 2 adjacent to a leaf of C . Each subdivision vertex adds a total of 2π to the both sides of the inequality, hence the contradiction holds. \square

A.9 Proof of Theorem 7

Theorem 7 There exists a set of six points in the plane whose Delaunay triangulation is not an increasing-chord drawing.

We present a counterexample in Figure 13. We have two circles of the same radius with the one on the right with its center slightly below the one on the left. The chords AB and $A'B'$ are vertical and the points C and C' are chosen such that C does not lie inside the circle on the right and C' does not lie inside the circle on the left. The triangles ABC and $A'B'C'$ must exist in the Delaunay triangulation. A simple case analysis then shows that there is no increasing chord path from B to B' in the Delaunay triangulation of the set of points shown (an increasing chord path cannot take any of the edges $AB, A'B', CC', AC$ or $A'C'$). Note that $BCA'B'$ is a self-approaching path.

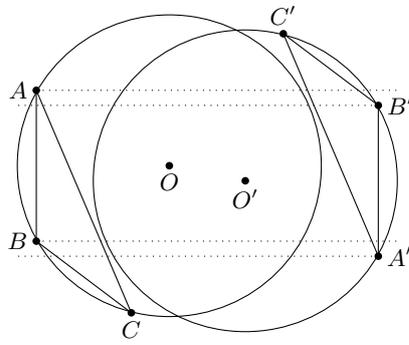


Fig. 13. A Delaunay triangulation that does not contain increasing-chord paths between all pairs of vertices.