Approximating geometric bottleneck shortest paths*

Prosenjit Bose[1], Anil Maheshwari[1], Giri Narasimhan[2], Michiel Smid[1], and Norbert Zeh[3]

[1] School of Computer Science, Carleton University, Ottawa, Canada K1S 5B6. {jit,maheshwa,michiel}@scs.carleton.ca
[2] School of Computer Science, Florida International University, Miami, FL 33199. giri@cs.fiu.edu
[3] Department of Computer Science, Duke University, Durham, NC 27708-0129. nzeh@cs.duke.edu

Abstract. In a geometric bottleneck shortest path problem, we are given a set S of n points in the plane, and want to answer queries of the following type: Given two points p and q of S and a real number L, compute (or approximate) a shortest path in the subgraph of the complete graph on S consisting of all edges whose length is less than or equal to L. We present efficient algorithms for answering several query problems of this type. Our solutions are based on minimum spanning trees, spanners, the Delaunay triangulation, and planar separators.

1 Introduction

We consider *bottleneck shortest path* problems in geometric graphs. Given a set S of n points in the plane, we consider queries of the following type: Given any two points p and q of S and any real number L, compute or approximate a shortest path in the subgraph of the complete graph on S consisting of all edges whose length is less than or equal to L.

To define these problems more precisely, given $L \in \mathbb{R}$, let $K^{(\leq L)}$ be the graph with vertex set S, in which any two distinct vertices p and q are connected by an edge if and only if their Euclidean distance $|pq|$ is less than or equal to L. Furthermore, we denote by $\delta^{(\leq L)}(p, q)$ the Euclidean length of a shortest path between p and q in the graph $K^{(\leq L)}$.

In a *bottleneck connectedness query*, we are given two points p and q of S and $L \in \mathbb{R}$, and have to decide if there exists a path between p and q in $K^{(\leq L)}$. In a *bottleneck shortest path length query*, we are given two points p and q of S and $L \in \mathbb{R}$, and have to compute $\delta^{(\leq L)}(p, q)$. In a *bottleneck shortest path query*, we are given two points p and q of S and $L \in \mathbb{R}$, and have to compute a path in $K^{(\leq L)}$ whose length is equal to $\delta^{(\leq L)}(p, q)$.

These problems are motivated by the following scenario. Imagine the points of S to be airports. Then we would like to answer queries in which we are given

* Bose, Maheshwari, and Smid were supported by NSERC.

two airports p and q and an airplane that can fly L miles without refueling, and have to compute shortest path information for this airplane to fly from p to q.

Throughout the rest of this paper, we denote by $L_1 < L_2 < \ldots < L_{\binom{n}{2}}$ the sorted sequence of distances determined by any two distinct points of S. (We assume for simplicity that all these distances are distinct.) For any i with $1 \leq i \leq \binom{n}{2}$, we write $K^{(i)}$ instead of $K^{(\leq L_i)}$, and $\delta^{(i)}(p,q)$ instead of $\delta^{(\leq L_i)}(p,q)$.

In Section 2, we show that, after an $O(n \log n)$–time preprocessing, bottleneck connectedness queries can be answered in $O(1)$ time. The data structure is a binary tree that reflects the way in which Kruskal's algorithm computes the minimum spanning tree of S.

In Section 3, we consider bottleneck shortest path length queries. We present a simple data structure of size $O(n^2 \log n)$ that supports ϵ-approximate bottleneck shortest path length queries in $O(\log n)$ time. A simple extension of this data structure allows ϵ-approximate bottleneck shortest path queries to be answered in $O(\log n + \ell)$ time, where ℓ is the number of edges on the reported path. This data structure uses $O(n^3 \log n)$ space.

In Section 4, we give a general approach for solving the approximate bottleneck shortest path query problem. The idea is to approximate the sequence $K^{(i)}$, $1 \leq i \leq \binom{n}{2}$, of graphs by a collection of $O(n)$ sparse graphs. We show that the bottleneck versions of the Yao-graph [7] and the Delaunay triangulation are examples of such collections of sparse graphs. The latter claim is obtained by extending the proof of Keil and Gutwin [5] that the Delaunay triangulation has stretch factor less than or equal to $2\pi/(3\cos(\pi/6))$. We prove that for any two points p and q of a given point set S, there exists a path in the Delaunay triangulation of S whose length is at most $2\pi/(3\cos(\pi/6)) \approx 2.42$ times the Euclidean distance $|pq|$ between p and q, and all of whose edges have length at most $|pq|$. (In [5], there is no guarantee on the lengths of the individual edges on the path.)

Finally, in Section 5, we give a data structure of size $O(n^{5/2})$ that can be used to answer bottleneck shortest path queries in planar graphs in $O(\sqrt{n} + \ell)$ time, where ℓ is the number of edges on the reported path. This data structure uses a result of Djidjev [3] to obtain a recursive separator decomposition of the planar graph. By applying this result to those of Section 4, we obtain an efficient solution to the approximate bottleneck shortest path query problem.

To our knowledge, the type of bottleneck shortest path problems considered in this paper have not been considered before. There is related work by Narasimhan and Smid [6], who consider the following problem: Given a real number L, approximate the *stretch factor* of the graph $K^{(\leq L)}$, which is defined as the maximum value of $\delta^{(\leq L)}(p,q)/|pq|$ over all distinct points p and q of S. They present a data structure of size $O(\log n)$, that can be built in roughly $O(n^{4/3})$ time, and that can be used to answer approximate stretch factor queries (with an approximation factor of about 36) in $O(\log \log n)$ time.

Our results are based on *t-spanners*, which are sparse graphs having stretch factor less than or equal to t; see the survey by Eppstein [4].

2 Bottleneck connectedness queries

Let $MST(S)$ be the Euclidean minimum spanning tree of the point set S. We define a binary tree $T(S)$ as follows. If $|S| = 1$, then $T(S)$ consists of one node storing the only point of S. Assume that $|S| \geq 2$, and let e be the longest edge in $MST(S)$. Removing e partitions $MST(S)$ into two trees. Let S_1 and S_2 be the vertex sets of these trees. Then $T(S)$ consists of a root that stores the edge e and pointers to its two children, which are roots of recursively defined trees $T(S_1)$ and $T(S_2)$. Observe that the leaves of $T(S)$ are in one-to-one correspondence with the points of S, and the internal nodes are in one-to-one correspondence with the edges of $MST(S)$. Computing $T(S)$ according to the above definition corresponds to tracing back the execution of Kruskal's minimum spanning tree algorithm. The following lemma shows how the tree $T(S)$ can be used to answer bottleneck connectedness queries.

Lemma 1. *Let p and q be two distinct points of S, and let L be a real number. Let e be the edge stored at the lowest common ancestor (LCA) of the leaves of $T(S)$ storing p and q. Then p and q are connected by a path in the graph $K^{(\leq L)}$ if and only if the length of e is less than or equal to L.*

Proof. Assume that the length of e is less than or equal to L. Let u be the node of $T(S)$ that stores e. We may assume w.l.o.g. that p is stored in the left subtree of u (and, hence, that q is stored in the right subtree of u). Let S_p and S_q be the sets of points that are stored at the leaves of the left and right subtrees of u, respectively. Let x and y be the endpoints of e, where $x \in S_p$ and $y \in S_q$. Consider what happens when Kruskal's algorithm computes the minimum spanning tree of S. Immediately before e is added, all points in S_p are connected by edges in the partially constructed minimum spanning tree, and the length of each of these edges is less than the length of e. The same holds for the set S_q. Hence, immediately after edge e has been added to the partially constructed minimum spanning tree, p and q are connected by a path, all of whose edges have a length that is at most the length of e. This path is contained in the graph $K^{(\leq L)}$. To prove the converse, assume that the length of e is larger than L. Let S_1 and S_2 be the partition of S obtained by deleting e from $MST(S)$. Since the unique path in $MST(S)$ between p and q contains e, we have (i) $p \in S_1$ and $q \in S_2$, or (ii) $p \in S_2$ and $q \in S_1$. By a well known property of minimum spanning trees, the length of e is equal to the minimum distance between any point in S_1 and any point of S_2. If there is a path in $K^{(\leq L)}$ between p and q, then this path must contain an edge between some point of S_1 and some point of S_2. Since the length of any such edge is larger than L, it follows that such a path cannot exist. $\square$

Lemma 1 implies that a bottleneck connectedness query can be answered by answering an LCA-query in the tree $T(S)$. This tree can be computed in $O(n \log n)$ time. Given $T(S)$, we preprocess it in $O(n)$ time, so that LCA-queries can be answered in $O(1)$ time. (See Bender and Farach-Colton [1].)

Theorem 1. *We can preprocess a set of n points in the plane in $O(n \log n)$ time into a data structure of size $O(n)$, such that bottleneck connectedness queries can be answered in $O(1)$ time.*

3 Bottleneck shortest path length queries

Recall that $K^{(i)}$ denotes the graph $K^{(\leq L_i)}$. We define $K^{(0)}$ to be the graph $(S, \emptyset)$. Also recall that we write $\delta^{(i)}(p,q)$ instead of $\delta^{(\leq L_i)}(p,q)$.

Let $0 < \epsilon \leq 3$ be a fixed real constant. In this section, we show how to preprocess the points of S into a data structure of size $O(n^2 \log n)$, such that ϵ-approximate bottleneck shortest path length queries can be answered in $O(\log n)$ time. To be more precise, we show that a query of the following type can be answered in $O(\log \log n)$ time: Given two points p and q of S and an index $0 \leq i \leq \binom{n}{2}$, compute an ϵ-approximation to $\delta^{(i)}(p,q)$, i.e., a real number Δ, such that $\delta^{(i)}(p,q) \leq \Delta \leq (1+\epsilon) \cdot \delta^{(i)}(p,q)$. Using an additional amount of $O(n^2)$ space, we will extend this solution to solve general ϵ-approximate bottleneck shortest path length queries (in which an arbitrary real number L is part of the query, rather than the distance L_i) in $O(\log n)$ time. Our solution is based on an approach by Narasimhan and Smid [6].

We fix two distinct points p and q of S, and observe that

$$|pq| = \delta^{(\binom{n}{2})}(p,q) \leq \ldots \leq \delta^{(1)}(p,q) \leq \delta^{(0)}(p,q) = \infty.$$

Let $k := \min\{i \geq 0 : \delta^{(i)}(p,q) < \infty\}$. Since p and q are not connected by a path in the graph $K^{(k-1)}$, we have $|pq| > L_{k-1}$ and, hence, $|pq| \geq L_k$. On the other hand, since p and q are connected by a path in $K^{(k)}$, and since any such path contains at most $n-1$ edges, we have $\delta^{(k)}(p,q) \leq (n-1)L_k \leq (n-1)|pq|$. Based on this observation, we partition the set $\{k, k+1, \ldots, \binom{n}{2}\}$ into $O(\log n)$ subsets, in the following way. For any $0 \leq j \leq \log_{1+\epsilon/3}(n-1)$, let

$$I_{pq}^j := \left\{ i : k \leq i \leq \binom{n}{2} \text{ and } (1+\epsilon/3)^j |pq| \leq \delta^{(i)}(p,q) < (1+\epsilon/3)^{j+1}|pq| \right\}.$$

We store for each integer j for which $I_{pq}^j \neq \emptyset$, (i) a value ℓ_j, which is the smallest element of the set I_{pq}^j, and (ii) a value $\Delta^{(j)}(p,q)$ which is equal to $(1+\epsilon/3) \cdot \delta^{(\ell_j)}(p,q)$. Let us see how we can use this information to answer an ϵ-approximate bottleneck shortest path length query for p and q. Let i be an integer with $0 \leq i \leq \binom{n}{2}$. We start by showing how the value of $\delta^{(i)}(p,q)$ can be approximated. First compute the integer j for which $\ell_j \leq i < \ell_{j-1}$. Then return the value $\Delta := \Delta^{(j)}(p,q)$.

To prove the correctness of this query algorithm, first observe that $i \in I_{pq}^j$. This implies that $\delta^{(i)}(p,q) < (1+\epsilon/3)^{j+1}|pq|$. Similarly, since $\ell_j \in I_{pq}^j$, we have $\delta^{(\ell_j)}(p,q) \geq (1+\epsilon/3)^j |pq|$. It follows that $\delta^{(i)}(p,q) \leq \Delta$. In a completely symmetric way, we obtain $\Delta \leq (1+\epsilon/3)^2 \cdot \delta^{(i)}(p,q)$, which is at most $(1+\epsilon) \cdot \delta^{(i)}(p,q)$. This proves that Δ is an ϵ-approximation to the length of a shortest path between p and q in the graph $K^{(i)}$. By storing the values ℓ_j in sorted order in an array, Δ can be computed in $O(\log \log n)$ time.

We store all this information for each pair of points. Additionally, we store the sequence $L_1 < L_2 < \ldots < L_{\binom{n}{2}}$ of distances. Given two query points p and q of S and an arbitrary query value $L \in \mathbb{R}$, we first use binary search to find the index i for which $L_i \leq L < L_{i+1}$. Since $\delta^{(\leq L)}(p,q) = \delta^{(i)}(p,q)$, we then answer the query as described above.

The amount of space used by this solution is $O(n^2 \log n)$, and the query time is $O(\log n)$. Let us consider the preprocessing time. It clearly suffices to solve the all-pairs-shortest-path problem for each graph $K^{(i)}$, $0 \leq i \leq \binom{n}{2}$. Using the Floyd-Warshall algorithm, one such problem can be solved in $O(n^3)$ time. Hence, the overall preprocessing time is $O(n^5)$.

If we store with each value $\Delta^{(j)}(p,q)$ a path in $K^{(\ell_j)}$ having length $\delta^{(\ell_j)}(p,q)$, then we can use this additional information to answer approximate bottleneck shortest path queries: Let L and i be as above. Then the path P stored with $\Delta^{(j)}(p,q)$ has length $\delta := \delta^{(\ell_j)}(p,q)$ satisfying $\delta^{(i)}(p,q) \leq \delta \leq (1+\epsilon/3)\delta^{(i)}(p,q)$. (Observe that P is a path in $K^{(i)}$.)

Theorem 2. *For any real constant $\epsilon > 0$, we can preprocess a set of n points in the plane in $O(n^5)$ time into*

1. *a data structure of size $O(n^2 \log n)$, such that ϵ-approximate bottleneck shortest path length queries can be answered in $O(\log n)$ time,*
2. *a data structure of size $O(n^3 \log n)$, such that ϵ-approximate bottleneck shortest path queries can be answered in $O(\log n + \ell)$ time, where ℓ is the number of edges on the reported path.*

4 The bottleneck shortest path problem

In this section, we introduce a general approach for the approximate bottleneck shortest path problem. The idea is to approximate the sequence $K^{(i)}$, $1 \leq i \leq \binom{n}{2}$, by a "small" collection of sparse graphs, i.e., with "few" edges. This notion is formalized in the definition below. For any graph G and any two vertices p and q, we denote the length of a shortest path in G between p and q by $\delta^{(G)}(p,q)$.

Definition 1. *Let S be a set of n points in the plane, let $t \geq 1$ be a real number, let J be a subset of $\{1, 2, \ldots, \binom{n}{2}\}$ and, for each $j \in J$, let $G^{(j)}$ be a graph with vertex set S all of whose edges have length at most L_j. We say that the collection $\mathcal{G} = \{G^{(j)} : j \in J\}$ is a* collective bottleneck t-spanner *of S, if the following holds: for any i with $1 \leq i \leq \binom{n}{2}$, there is an index $j \in J$, such that $j \leq i$ and*

$$\delta^{(i)}(p,q) \leq \delta^{(G^{(j)})}(p,q) \leq t \cdot \delta^{(i)}(p,q)$$

holds for all pairs of points p and q in S.

In order to approximate a bottleneck shortest path between p and q in the possibly dense graph $K^{(i)}$, we compute a shortest path P between p and q in the graph $G^{(j)}$. Since $j \leq i$, P is a path in $K^{(i)}$; it is a t-approximate shortest path between p and q in $K^{(i)}$.

The goal is to define $\mathcal{G}$ in such a way that shortest path queries on them can be answered efficiently. Further goals are to minimize (i) the value of t, (ii) the size of the index set J, and (iii) the number of edges in the graphs in $\mathcal{G}$.

Lemma 2. *The size of the index set J is greater than or equal to $n-1$.*

Proof. Let (p,q) be any edge of the Euclidean minimum spanning tree $MST(S)$ of S, and let i be the index such that $|pq| = L_i$. Observe that $\delta^{(i)}(p,q) = |pq| < \infty$. We claim that $i \in J$. To prove this, assume that $i \notin J$. By Definition 1, there is an index $j \in J$ such that $j < i$ and $\delta^{(G^j)}(p,q) \leq t \cdot \delta^{(i)}(p,q) < \infty$. In particular, we have $\delta^{(j)}(p,q) < \infty$. By well known properties of minimum spanning trees, however, $\delta^{(j)}(p,q) = \infty$, contradicting our assumption that $i \notin J$. Hence, each edge of $MST(S)$ contributes an index to J. □

In the rest of this section, we give some examples of collective bottleneck spanners.

The Yao-graph [7]: Let S be a set of n points in the plane, and let θ be an angle such that $2\pi/\theta$ is an integer. We partition the plane into a collection $\mathcal{C}$ of $2\pi/\theta$ cones of angle θ, all having their apex at the origin. For any point $p \in S$ and any cone $C \in \mathcal{C}$, let C_p be the cone obtained by translating C by the vector $\boldsymbol{p}$. (Hence, C_p has p as its apex.)

The Yao-graph $Y(S,\theta)$ has S as its vertex set. Let p be any point of S, let C be any cone of $\mathcal{C}$ such that $C_p \cap (S \setminus \{p\}) \neq \emptyset$, and let q_p be the point of $C_p \cap (S \setminus \{p\})$ whose Euclidean distance to p is minimum. The edge set of $Y(S,\theta)$ consists of all edges (p,q_p) obtained in this way. Chang *et al.* [2] have shown how to construct the graph $Y(S,\theta)$ in $O(n \log n)$ time.

Given two points p and q, we construct a path between p and q in $Y(S,\theta)$ in the following way. If $p = q$, then there is nothing to do. Assume that $p \neq q$. Let C be the cone in $\mathcal{C}$ such that $q \in C_p$. The graph $Y(S,\theta)$ contains an edge (p,r), where $r \in C_p$ and $|pr| \leq |pq|$. We follow this edge, and recursively construct a path between r and q. It can be shown that for any constant $\epsilon > 0$, there is an angle θ such that the path constructed in this way has length at most $(1+\epsilon)|pq|$ and the length of each edge on this path is less than or equal to $|pq|$; see [5].

If we denote the number of edges of $Y(S,\theta)$ by m, then $m = O(n)$. Let $j_1 < j_2 < \ldots < j_m$ be the indices such that $L_{j_1} < L_{j_2} < \ldots < L_{j_m}$ are the edge lengths of $Y(S,\theta)$. For any k with $1 \leq k \leq m$, denote by $Y^{(j_k)}$ the graph consisting of all edges of $Y(S,\theta)$ whose lengths are at most L_{j_k}.

Let $J := \{j_k : 1 \leq k \leq m\}$. We claim that $\mathcal{G} := \{Y^{(j_k)} : 1 \leq k \leq m\}$ is a collective bottleneck $(1+\epsilon)$-spanner of S. To prove this claim, consider any two points p and q of S and any integer $1 \leq i \leq \binom{n}{2}$. We may assume that p and q are connected by a path in $K^{(i)}$. Let k be the integer such that $L_{j_k} \leq L_i < L_{j_{k+1}}$, and let δ be the length of a shortest path between p and q in the graph $Y^{(j_k)}$. It is clear that $j_k \leq i$. It remains to show that $\delta^{(i)}(p,q) \leq \delta \leq (1+\epsilon) \cdot \delta^{(i)}(p,q)$. The first inequality follows from the fact that $j_k \leq i$. This implies that $L_{j_k} \leq L_i$ and, therefore, $Y^{(j_k)}$ is a subgraph of $K^{(i)}$. To prove the second inequality, consider a shortest path P between p and q in $K^{(i)}$. Hence, the length of P is equal to $\delta^{(i)}(p,q)$. Consider an arbitrary edge (x,y) on P. Observe that $|xy| \leq L_i$. In the graph $Y(S,\theta)$, there is a path P'_{xy} between x and y whose length is at most $(1+\epsilon)|xy|$ and all of whose edges are of length at most $|xy|$. Hence, P'_{xy} is a path in the graph $Y^{(j_k)}$. By concatenating the paths P'_{xy}, over all edges (x,y) of P, we obtain a path between p and q in $Y^{(j_k)}$ having length at most $(1+\epsilon)$ times the length of P. This proves the second inequality.

Theorem 3. *Let S be a set of n points in the plane, and let $\epsilon > 0$ be a constant. There exists a collective bottleneck $(1+\epsilon)$-spanner of S, consisting of $O(n)$ graphs.*

Observe that the bottleneck graphs in the collection $\mathcal{G}$ are not planar (except for small values of k). We next show how to obtain a collective bottleneck spanner where the collection consists of $O(n)$ planar graphs.

The Delaunay triangulation: Let S be a set of n points in the plane, and consider the Delaunay triangulation $DT(S)$ of S. In this section, we show that for any two points p and q of S, there exists a path $P_{DT(S)}(p,q)$ between p and q in $DT(S)$ such that (i) no edge on this path has length more than $|pq|$ and (ii) the length of $P_{DT(S)}(p,q)$ is less than or equal to $2\pi/(3\cos(\pi/6)) \cdot |pq|$. We will prove this claim by modifying Keil and Gutwin's proof of the fact that $DT(S)$ is a $(2\pi/(3\cos(\pi/6))$-spanner of S; see [5]. (The proof in [5] may produce a path between p and q that contains an edge whose length is larger than $|pq|$.) We start with the following key lemma, which is similar to Lemma 1 in [5].

Lemma 3. *Let p and q be two points of S on a horizontal line L such that p is to the left of q. Let C be a circle having p and q on its boundary, let r and O be the radius and center of C, respectively, and let θ be the upper angle $\angle pOq$. Assume that no point of S lies inside C below L. Then there exists a path $P_{DT(S)}(p,q)$ between p and q in $DT(S)$ such that (i) no edge on this path has length more than $|pq|$, and (ii) the length of $P_{DT(S)}(p,q)$ is less than or equal to $r\theta$.*

Proof. We may assume w.l.o.g. that p and q are on the x-axis. If the circle C contains no points of S in its interior, then the lemma holds since by the empty circle property of Delaunay triangulations, (p,q) is an edge of $DT(S)$.

In the rest of the proof, we assume that C contains at least one point of S in its interior (implying that (p,q) is not an edge of $DT(S)$). Observe that, by the assumptions in the lemma, all points of $C \cap S$ must be above the x-axis.

Our goal is to prove the lemma by induction on the angle θ. In order to do this, we will *normalize* the circle C so that, over all possible pairs p and q of points in S, there are only a finite number of normalized circles.

We move the center O of C downwards along the bisector of p and q. During this movement, we change C so that the points p and q stay on its boundary. (Hence, the radius r and the upper angle θ of C change. It is easy to see that $r\theta$ decreases.) We stop this process at the moment when the part of C below the

x-axis hits at a point, say z, of S. Observe that z exists, because otherwise (p, q) is an edge of the convex hull of S, and thus an edge of $DT(S)$, which would contradict our earlier assumption.

We refer to this circle as a normalized circle. Hence, from now on, C is a normalized circle with center O and radius r, containing p, q, and z on its boundary, where z is below the x-axis. There are no points of S in the interior of C below the x-axis, and there is at least one point of S in the interior of C.

Each pair of points defines at most two normalized circles (depending on whether the circle is empty above or below the line through the two points). Therefore, there are at most $O(n^2)$ possible normalized circles. We proceed by induction on the rank of the angle θ of these circles.

For the base case, consider the circle C with the smallest angle θ. For any point t of $C \cap S$, let D_t be the circle through p, q, and t. Choose the t for which no point of S lies in the interior of D_t above the x-axis. (Observe that such a t must exist.) We will write D instead of D_t.

Let C_1 be the circle through p and t whose center lies on the segment pO. We denote the center of C_1 by O_1. By the definition of D, no point of S lies in C_1 below the line through pt. However, our construction implies that $\angle pO_1t < \angle pOq$, which contradicts the assumption that θ is the smallest angle. We conclude that the circle C must be empty and, hence, (p, q) is an edge of $DT(S)$. Thus, $|pq| \leq r\theta$ as required. This proves the base case of the induction.

For the inductive step, we consider a given normalized circle C and assume that the lemma holds for all normalized circles with smaller interior upper angle. Define the point t and circle D as above. Observe that the upper angle $\theta = \angle pOq$ lies in the open interval $(0, 2\pi)$.

We will treat the cases $0 < \theta \leq \pi$ and $\pi < \theta < 2\pi$ separately. Before proceeding, we outline a few constructions which will be useful in the sequel.

Let C_1 be the circle through p and t whose center lies on the segment pO. We denote the center and radius of C_1 by O_1 and r_1, respectively. Similarly, let C_2 be the circle through t and q whose center lies on the segment qO. We denote the center and radius of C_2 by O_2 and r_2, respectively. Finally, circle D implies that no point of S lies in C_1 below the line through p and t, and no point of S lies in C_2 below the line through q and t.

Consider the two intersection points between C_1 and the x-axis. One of these intersection points is p; we denote the other one by a_1. Similarly, let a_2 be the intersection point between C_2 and the x-axis that is not equal to q.

Let C_3 be the circle through a_1 and a_2 whose center is the intersection between the line through O_1 and a_1 and the line through O_2 and a_2. We denote the center and radius of C_3 by O_3 and r_3, respectively.

We observe that the following four triangles are all similar isoceles triangles with two equal base angles, denoted by ϕ: $\triangle(p, O, q)$, $\triangle(p, O_1, a_1)$, $\triangle(a_2, O_2, q)$, and $\triangle(a_2, O_3, a_1)$.

By construction, θ is larger than both angles $\theta_1 = \angle pO_1t$ and $\theta_2 = \angle tO_2q$. Therefore, by the induction hypothesis, there exists a path $P_{DT(S)}(p, t)$ between p and t in $DT(S)$, all of whose edges have length at most $|pt|$, and $|P_{DT(S)}(p, t)| \leq$

$r_1\theta_1$. Similarly, there exists a path $P_{DT(S)}(t,q)$ between t and q in $DT(S)$, all of whose edges have length at most $|tq|$, and $|P_{DT(S)}(t,q)| \leq r_2\theta_2$.

Case 1: $0 < \theta \leq \pi$. Let $P_{DT(S)}(p,q)$ be the concatenation of the paths $P_{DT(S)}(p,t)$ and $P_{DT(S)}(t,q)$. We have

$$\begin{aligned}|P_{DT(S)}(p,q)| &= |P_{DT(S)}(p,t)| + |P_{DT(S)}(t,q)| \\ &\leq r_1\theta_1 + r_2\theta_2 \\ &= r_1\theta + r_2\theta - [r_1(\theta - \theta_1) + r_2(\theta - \theta_2)].\end{aligned}$$

Since $0 < \theta \leq \pi$, the point a_2 is to the left of a_1. Therefore, the part of C_3 that is above the x-axis is contained in the part of $C_1 \cap C_2$ that is above the x-axis. This, in turn, implies that $r_1(\theta - \theta_1) + r_2(\theta - \theta_2) \geq r_3\theta$. Hence, we have $|P_{DT(S)}(p,q)| \leq (r_1 + r_2 - r_3)\theta$. Finally, a straightforward analysis yields

$$\begin{aligned}|P_{DT(S)}(p,q)| &\leq (r_1 + r_2 - r_3)\theta \\ &= (|pa_1|/2\cos\phi + |a_2q|/2\cos\phi - |a_2a_1|/2\cos\phi) \\ &= |pq|/2\cos\phi = r\theta.\end{aligned}$$

Since $0 < \theta \leq \pi$, we have $|pt| \leq |pq|$ and $|tq| \leq |pq|$. Therefore, the length of each edge on $P_{DT(S)}(p,q)|$ is less than or equal to $|pq|$.

Case 2.a: $\pi < \theta < 2\pi$ and the point t is inside the circle having p and q as diameter. Let $P_{DT(S)}(p,q)$ be the concatenation of the paths $P_{DT(S)}(p,t)$ and $P_{DT(S)}(t,q)$. We have $|P_{DT(S)}(p,q)| \leq |P_{DT(S)}(p,t)| + |P_{DT(S)}(t,q)| \leq r_1\theta_1 + r_2\theta_2$. If the point a_2 is to the right of a_1, then θ_1 and θ_2 are both less than or equal to θ. Therefore, we have $|P_{DT(S)}(p,q)| \leq (r_1 + r_2)\theta$. A straightforward analysis gives

$$\begin{aligned}|P_{DT(S)}(p,q)| &\leq (r_1 + r_2)\theta \\ &= (|pa_1|/2\cos\phi + |a_2q|/2\cos\phi) \\ &\leq |pq|/2\cos\phi = r\theta.\end{aligned}$$

If a_2 is to the left of a_1, then $|P_{DT(S)}(p,q)| \leq r\theta$ by the same argument as in Case 1.

Since t is contained in the circle with p and q as diameter, both $|pt|$ and $|tq|$ are less than $|pq|$. As a result, the length of each edge on $P_{DT(S)}(p,q)$ is less than or equal to $|pq|$.

Case 2.b: $\pi < \theta < 2\pi$ and the point t lies outside the circle R with p and q as diameter. Let C_4 be the circle through p and z whose center is on the x-axis. We denote the center and radius of C_4 by O_4 and r_4, respectively. Let θ_4 be the lower angle $\angle pO_4z$. Similarly, let C_5 be the circle through q and z whose center is on the x-axis. We denote the center and radius of C_5 by O_5 and r_5, respectively. Let θ_5 be the lower angle $\angle zO_5q$.

The fact that no point of S is contained in the part of R that is above the x-axis, implies that there are no points of S in the part of C_4 that is above the line through p and z. Similarly, there are no points of S in the part of C_5 that is above the line through q and z. Furthermore, since both angles θ_4 and θ_5 are less than π and, hence, less than θ, we can apply the induction hypothesis. That is, there exists a path $P_{DT(S)}(p,z)$ between p and z in $DT(S)$, all of whose edges have length at most $|pz|$, and $|P_{DT(S)}(p,z)| \leq r_4\theta_4$. Similarly, there exists a path $P_{DT(S)}(z,q)$ between z and q in $DT(S)$, all of whose edges have length at most $|zq|$, and $|P_{DT(S)}(z,q)| \leq r_5\theta_5$.

Let $P_{DT(S)}(p,q)$ be the concatenation of $P_{DT(S)}(p,z)$ and $P_{DT(S)}(z,q)$. Then

$$\begin{aligned} |P_{DT(S)}(p,q)| &\leq |P_{DT(S)}(p,z)| + |P_{DT(S)}(z,q)| \\ &\leq r_4\theta_4 + r_5\theta_5 \leq (r_4+r_5)\theta \leq r\theta, \end{aligned}$$

where the last inequality follows from the fact that both r_4 and r_5 are less than or equal to $|pq|/2$. Finally, since both $|pz|$ and $|zq|$ are less than $|pq|$, the length of each edge on $P_{DT(S)}(p,q)|$ is less than or equal to $|pq|$. □

Theorem 4. *Let S be a set of n points in the plane, let $DT(S)$ be the Delaunay triangulation of S, and let p and q be two points of S. There is a path between p and q in $DT(S)$ whose length is less than or equal to $2\pi/(3\cos(\pi/6)) \cdot |pq|$, and all of whose edges have length at most $|pq|$.*

Proof. The proof is similar to the proof of Theorem 1 in [5]. □

Let m be the number of edges of $DT(S)$, and let $j_1 < j_2 < \ldots < j_m$ be the indices such that $L_{j_1} < L_{j_2} < \ldots < L_{j_m}$ are the edge lengths of $DT(S)$. Observe that $m \leq 3n-6$. For any k with $1 \leq k \leq m$, let $DT^{(j_k)}$ be the graph with vertex set S consisting of all edges of $DT(S)$ whose lengths are at most L_{j_k}.

Corollary 1. *The collection $\mathcal{G} := \{DT^{(j_k)} : 1 \leq k \leq m\}$ of planar graphs constitutes a collective bottleneck t-spanner of S for $t = 2\pi/(3\cos(\pi/6))$.*

5 Bottleneck shortest path queries in planar graphs

In this section, we address the following problem: Given a set S of n points in the plane and a planar graph G with vertex set S, build a data structure that can answer bottleneck queries of the following type: Given two points p and q of S and a real number L, decide whether there is a path between p and q in G all of whose edges are of length at most L, and report the shortest such path if such a path exists. We will present a structure of size $O(n^{5/2})$ that answers existence queries in $O(\log n)$ time and allows the shortest path whose edges have length at most L to be reported in $O(\sqrt{n}+\ell)$ time, where ℓ is the number of edges on the reported path. We start in Section 5.1 with reviewing Djidjev's data structure [3] for answering general shortest path queries in the planar graph G. In Section 5.2, we show how to answer bottleneck shortest path queries in G.

5.1 Shortest path queries in G

Djidjev's shortest path data structure [3] is based on a recursive separator partition of the planar graph G, which we represent by a *separator tree* $\mathcal{T}(G)$. This tree is defined as follows. If G has only one vertex p, then $\mathcal{T}(G)$ contains only one node, say α. We store with α the set $S(\alpha) := \{p\}$, and associate with α the graph $G(\alpha) := G$. Otherwise, we compute a set C of $O(\sqrt{n})$ vertices in G such that each of the connected components $G_1, \ldots, G_k$ of $G \setminus C$ has size at most $2n/3$. We store the set $S(\alpha) := C$ with the root α of $\mathcal{T}(G)$, and associate the graph $G(\alpha) := G$ with α. Then we recursively compute separator trees $\mathcal{T}(G_1), \ldots, \mathcal{T}(G_k)$ for the graphs $G_1, \ldots, G_k$, and make the roots of those trees the children of α.

It is easy to see that each point of S is stored exactly once in $\mathcal{T}(G)$, so that we can associate a unique node $\alpha(p)$ of $\mathcal{T}(G)$ with every point p of S.

For any two points p and q of S, every path in G between p and q must contain at least one vertex stored at a common ancestor of $\alpha(p)$ and $\alpha(q)$. Thus, we associate a set $A(p)$ with every point p of S. This set $A(p)$ consists of all pairs $(x, \delta^{(G(\alpha(x)))}(p, x))$, where x ranges over all points of S that are stored at some ancestor of $\alpha(p)$ in the separator tree $\mathcal{T}(G)$, and $\delta^{(G(\alpha(x)))}(p, x)$ is the length of a shortest path between p and x in $G(\alpha(x))$. Each such set $A(p)$ can be shown to have size $O(\sqrt{n})$, so that the total size of all sets $A(p)$, $p \in S$, is $O(n^{3/2})$. Since every path between two points p and q must contain at least one vertex stored at an ancestor of $\alpha(p)$ and $\alpha(q)$, the value of $\delta^{(G)}(p, q)$ is equal to the minimum value of $\delta^{(G(\alpha(x)))}(p, x) + \delta^{(G(\alpha(x)))}(q, x)$, where x ranges over all vertices for which $(x, \delta^{(G(\alpha(x)))}(p, x)) \in A(p)$ and $(x, \delta^{(G(\alpha(x)))}(q, x)) \in A(q)$.

Thus, we can answer a shortest path-length query in G between two vertices p and q in $O(\sqrt{n})$ time by scanning the lists $A(p)$ and $A(q)$. In [3], it is shown how to augment the data structure with shortest path trees rooted at the separator vertices, so that the actual shortest path between p and q can be reported in $O(\ell)$ additional time, where ℓ is the number of edges on the reported path.

Lemma 4 ([3]). *Let S be a set of n points in the plane and let G be a planar graph with vertex set S. We can preprocess G in $O(n^{3/2})$ time into a data structure of size $O(n^{3/2})$ such that the shortest path in G between any two query points can be computed in $O(\sqrt{n} + \ell)$ time, where ℓ is the number of edges on the reported path.*

5.2 Bottleneck shortest path queries in G

Consider again the planar graph G with vertex set S. Let $e_1, e_2, \ldots, e_m$ be the m edges of G, sorted by their lengths. For any i with $1 \leq i \leq m$, let $|e_i|$ denote the Euclidean length of edge e_i, and let $G^{(i)}$ be the graph consisting of all edges of G having length at most $|e_i|$.

In order to answer bottleneck shortest path queries in G, for each of the graphs $G^{(i)}$, we build the shortest path data structure of Lemma 4. We also compute a labeling of the vertices of each $G^{(i)}$ so that two vertices have the same label if and only if they are in the same connected component of $G^{(i)}$.

Observation 1 *Let p and q be two points of S, let L be a real number, and let i be the integer such that $|e_i| \leq L < |e_{i+1}|$. There is a path between p and q in G all of whose edges have length at most L if and only if p and q are in the same connected component of $G^{(i)}$. Furthermore, the shortest path between p and q in*

$G^{(i)}$ *is the same as the shortest path between* p *and* q *in* G *all of whose edges have length at most* L.

Thus, we build a binary search tree T over the sorted edge set of G. In $O(\log n)$ time, we can find the index i such that $|e_i| \leq L < |e_{i+1}|$. Given that every node of T stores a pointer to the corresponding graph $G^{(i)}$ and the shortest path data structure for $G^{(i)}$, it now takes constant time to retrieve the two labels of the vertices p and q in $G^{(i)}$, and compare them to decide whether p and q are in the same connected component of $G^{(i)}$. If they are, we query the shortest path data structure to report the shortest path, which takes $O(\sqrt{n} + \ell)$ time, where ℓ is the number of edges in the reported path, using the data structure of Lemma 4. Since G is planar, we build $O(n)$ shortest path data structures of size $O(n^{3/2})$ each, one per graph $G^{(i)}$. Each of these data structures can be constructed in $O(n^{3/2})$ time. Hence, the total preprocessing time and amount of space used by our data structure is $O(n^{5/2})$.

Theorem 5. *Let* S *be a set of* n *points in the plane and let* G *be a planar graph with vertex set* S. *We can preprocess* G *in* $O(n^{5/2})$ *time into a data structure of size* $O(n^{5/2})$ *such that the following type of bottleneck queries can be answered: Given any two points* p *and* q *of* S *and any real number* L, *decide whether there is a path between* p *and* q *in* G *all of whose edges have length at most* L. *If such a path exists, report the shortest such path. The decision part of the query takes* $O(\log n)$ *time, whereas reporting the shortest path takes* $O(\sqrt{n} + \ell)$ *time, where* ℓ *is the number of edges on the reported path.*

If we combine Theorems 4 and 5, then we obtain the following result.

Theorem 6. *Let* S *be a set of* n *points in the plane. We can preprocess* S *in* $O(n^{5/2})$ *time into a data structure of size* $O(n^{5/2})$ *such that* t*-approximate bottleneck shortest path queries, for* $t = 2\pi/(3\cos(\pi/6))$, *can be answered in* $O(\sqrt{n} + \ell)$ *time, where* ℓ *is the number of edges on the reported path.*

References

1. M. A. Bender and M. Farach-Colton. The LCA problem revisited. *Proc. 4th LATIN*, LNCS, 1776, pp. 88–94, 2000.
2. M. S. Chang, N.-F. Huang, and C.-Y. Tang. An optimal algorithm for constructing oriented Voronoi diagrams and geographic neighborhood graphs. *Information Processing Letters*, 35:255–260, 1990.
3. H. N. Djidjev. Efficient algorithms for shortest path queries in planar digraphs. *Proc. 22nd WG*, LNCS, 1197, pp. 151–165, 1996.
4. D. Eppstein. Spanning trees and spanners. In *Handbook of Computational Geometry*, pages 425–461. Elsevier, 2000.
5. J. M. Keil and C. A. Gutwin. Classes of graphs which approximate the complete Euclidean graph. *Discrete & Computational Geometry*, 7:13–28, 1992.
6. G. Narasimhan and M. Smid. Approximation algorithms for the bottleneck stretch factor problem. *Nordic Journal of Computing*, 9:13–31, 2002.
7. A. C. Yao. On constructing minimum spanning trees in k-dimensional spaces and related problems. *SIAM Journal on Computing*, 11:721–736, 1982.