Approximating geometric bottleneck shortest paths*

Prosenjit Bose† Anil Maheshwari† Giri Narasimhan‡
Michiel Smid† Norbert Zeh§

February 18, 2004

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 between p and q in the subgraph of the complete graph on S consisting of all edges whose lengths are less than or equal to L. We present efficient algorithms for answering several query problems of this type. Our solutions are based on Euclidean minimum spanning trees, spanners, and the Delaunay triangulation. A result of independent interest is the following. For any two points p and q of S, there is a path between p and q in the Delaunay triangulation, whose length is less than or equal to $2\pi/(3\cos(\pi/6))$ times the Euclidean distance $|pq|$ between p and q, and all of whose edges have length at most $|pq|$.

*A preliminary version of this paper appeared in the Proceedings of the 20th Symposium on Theoretical Aspects of Computer Science (STACS), Lecture Notes in Computer Science, Vol. 2607, Springer-Verlag, Berlin, 2003, pp. 38–49.

†School of Computer Science, Carleton University, Ottawa, Ontario, Canada K1S 5B6. E-mail: {jit,maheshwa,michiel}@scs.carleton.ca. These authors were supported by NSERC.

‡School of Computer Science, Florida International University, Miami, FL 33199, USA. E-mail: giri@cs.fiu.edu.

§Faculty of Computer Science, Dalhousie University, Halifax, Nova Scotia, Canada B3H 1W5. E-mail: nzeh@cs.dal.ca.

1 Introduction

We consider *bottleneck shortest path* problems in geometric graphs. For 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 between p and q in the subgraph of the complete graph on S consisting of all edges whose lengths are 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 $K^{(\leq L)}$. (If there is no path between p and q in $K^{(\leq L)}$, then $\delta^{(\leq L)}(p, q) = \infty$.) In this paper, we will consider the following three query problems.

1. In a *bottleneck connectedness query*, we are given two points p and q of S and a real number L, and have to decide if there exists a path between p and q in $K^{(\leq L)}$.

2. In a *bottleneck shortest path length query*, we are given two points p and q of S and a real number L, and have to compute $\delta^{(\leq L)}(p, q)$ or an approximation to $\delta^{(\leq L)}(p, q)$.

3. In a *bottleneck shortest path query*, we are given two points p and q of S and a real number L, and have to compute a path between p and q in $K^{(\leq L)}$ whose length is equal to, or approximates, $\delta^{(\leq L)}(p, q)$.

The motivation for studying these problems comes from several applications. For example, consider a scenario where there are a number of wireless devices each with a specified radius of transmission L. Two devices p and q can communicate with each other if their distance is at most L. If their distance is more than L, then they can still communicate provided that there is a sequence of wireless devices each of whose distance is at most L from its neighbor in the sequence. This is precisely a path in $\delta^{(\leq L)}(p, q)$. Such a wireless network is referred to as an *ad-hoc wireless network*[15].

One can imagine another scenario where the points of S are airports. Then we would like to answer queries in which we are given two airports p and q and an airplane that can fly a distance of L kilometres without

refueling, and have to compute, or approximate, shortest path information for this airplane to fly from p to q.

Observe that there are $\binom{n}{2}$ pairs of points in S and $\binom{n}{2}$ graphs $K^{(\leq L)}$. This implies that the number of possible queries is $\Theta(n^4)$. As a result, both bottleneck connectedness queries and bottleneck shortest path length queries can trivially be solved in $O(\log n)$ time using $O(n^4)$ space. Similarly, using $O(n^5)$ space, bottleneck shortest path queries can trivially be solved in $O(\ell)$ time, where ℓ is the number of edges on the reported path.

1.1 Our results

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. By using the fact that $|pq| \leq \delta^{(i)}(p,q) \leq (n-1)|pq|$ for any i for which $\delta^{(i)}(p,q)$ is finite, 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, where ϵ is any fixed positive real constant. 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. Our approach is to approximate the sequence $K^{(i)}$, $1 \leq i \leq \binom{n}{2}$, of graphs by a collection of $O(n)$ sparse graphs. (A precise definition of this notion is given in Section 4.) Examples of such collections are given in Section 5. In Section 5.1, we show that the bottleneck version of the Yao-graph [16] is an example of such a collection of sparse graphs. Using the single-sink spanners of Arya *et al.* [1], we even obtain such a collection consisting of graphs of bounded degree (which are, in general, not planar). In Section 5.2, we prove that the bottleneck version of

the Delaunay triangulation gives such a collection, consisting of at most $3n-6$ planar graphs. The latter claim is obtained by extending the proof of Keil and Gutwin [10] that the Delaunay triangulation has stretch factor less than or equal to $2\pi/(3\cos(\pi/6))$. To be more precise, we prove that for any two points p and q of a given point set S, there exists a path between p and q 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 [10], there is no guarantee on the lengths of the individual edges on the path.)

Finally, in Section 6, 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 [7] to obtain a recursive separator decomposition of the planar graph. By applying this result to the graphs of Section 5.2, we obtain an efficient solution for the approximate bottleneck shortest path query problem.

1.2 Related results

After we wrote a preliminary version of this paper, we learned that the bottleneck connectedness query problem has been solved before, see Neto [12]. In fact, Neto's solution is identical to ours.

To the best of our knowledge, the other types of bottleneck shortest path problems considered in this paper have not been studied before. There is related work by Narasimhan and Smid [11], 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. A good overview of results on the problem of constructing t-spanners for a given point set can be found in the surveys by Eppstein [8] and Smid [14].

2 Bottleneck connectedness queries

As mentioned above, our solution for answering bottleneck connectedness queries appears already in Neto [12]. In order to be self-contained, however, we present this solution in this section.

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 [6]. It is not difficult to see that $T(S)$ can in fact be computed directly while running Kruskal's 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 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 without loss of generality 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 $y \in S_q$. By the recursive definition of $T(S)$, the edges of the subtree of $MST(S)$ induced by the points in S_p form a subtree of $MST(S)$. All edges in this subtree are of length at most that of e. Hence, there is a path P_1 in $MST(S)$ between p and x whose edges have length at most that of e. Similarly, there is a path P_2 in $MST(S)$ between y and q whose edges have length at most that of e. Thus, the path $P_1 \circ e \circ P_2$ is a path between p and q whose edges have length at most L. In particular, this path is contained in $K^{(\leq L)}$.

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 of 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. ∎

Lemma 1 implies that a bottleneck connectedness query can be answered by answering a lowest common ancestor query in the tree $T(S)$. This tree can be computed in $O(n \log n)$ time, by first computing the Delaunay triangulation $DT(S)$ of S (see [3]), and then running Kruskal's algorithm on $DT(S)$ (see [6]). Given $T(S)$, we preprocess it in $O(n)$ time, so that lowest common ancestor queries can be answered in $O(1)$ time. (See Harel and Tarjan [9], Schieber and Vishkin [13], or Bender and Farach-Colton [2].) We have proved the following result.

Theorem 2 *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 the sequence $L_1 < L_2 < \ldots < L_{\binom{n}{2}}$ of distances determined by any two distinct points of the point set S. Also, recall that, for $1 \leq i \leq \binom{n}{2}$, $K^{(i)}$ denotes the graph $K^{(\leq L_i)}$, i.e., the graph with vertex set S in which any two distinct points p and q are connected by an edge if and only if $|pq| \leq L_i$. We define $K^{(0)}$ to be the graph $(S, \emptyset)$. Finally, recall that we write $\delta^{(i)}(p, q)$ instead of $\delta^{(\leq L_i)}(p, q)$.

Let ϵ be any fixed real constant with $0 < \epsilon \leq 3$. 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. First 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 i with $0 \leq i \leq \binom{n}{2}$, compute an ϵ-approximation to the length $\delta^{(i)}(p, q)$

of a shortest path between p and q in the graph $K^{(i)}$, 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 [11].

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

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

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

$$\delta^{(k)}(p,q) \leq (n-1)L_k \leq (n-1)|pq|.$$

Hence, for all i with $k \leq i \leq \binom{n}{2}$, we have

$$|pq| \leq \delta^{(i)}(p,q) \leq (n-1)|pq|.$$

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 integer j, let

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

Clearly, I_{pq}^j can only be non-empty if $0 \leq j \leq \log_{1+\epsilon/3}(n-1)$. We store for each integer j where $I_{pq}^j \neq \emptyset$,

1. a value ℓ_{pq}^j, which is the smallest element of the set I_{pq}^j,
2. a value $\Delta^{(j)}(p,q)$ which is equal to $(1+\epsilon/3) \cdot \delta^{(\ell_{pq}^j)}(p,q)$.

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 $\ell_{pq}^j \leq i < \ell_{pq}^{j-1}$. 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 $\ell_{pq}^j \in I_{pq}^j$, we

have $\delta^{(\ell^j_{pq})}(p,q) \geq (1+\epsilon/3)^j|pq|$. By combining these two inequalities, it follows that $\delta^{(i)}(p,q) < \Delta$. In a completely symmetric way, we obtain

$$\Delta = (1+\epsilon/3)\cdot\delta^{(\ell^j_{pq})}(p,q) < (1+\epsilon/3)^{j+2}|pq| \leq (1+\epsilon/3)^2\cdot\delta^{(i)}(p,q).$$

Since $0 < \epsilon \leq 3$, we have $(1+\epsilon/3)^2 \leq 1+\epsilon$. Therefore, $\Delta < (1+\epsilon)\cdot\delta^{(i)}(p,q)$. 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_{pq} in sorted order in an array, the ϵ-approximation to $\delta^{(i)}(p,q)$ can be computed in $O(\log\log n)$ time.

We store 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 $O(n^2\log n)$, because we store $O(\log n)$ values for each pair of points of S. Furthermore, 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; see [6]. Hence, the overall preprocessing time is $O(n^5)$.

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

Theorem 3 *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 $K^{(i)}$, $1 \leq i \leq \binom{n}{2}$, of graphs 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*

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

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

The purpose of this definition should be clear: 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 $G^{(j)}$. Observe that, since $j \leq i$, P is a path in $K^{(i)}$ and $\delta^{(i)}(p,q) \leq \delta^{(G^{(j)})}(p,q)$. Hence, P is a t-approximate shortest path between p and q in $K^{(i)}$.

The goal is to define the collection $\mathcal{G}$ 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 $\mathcal{G}$. The following lemma gives a lower bound on the size of J.

Lemma 4 *The size of the index set J in Definition 1 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 (see also the proof of Lemma 1),

however, we have $\delta^{(j)}(p,q) = \infty$, contradicting our assumption that $i \notin J$. Hence, each of the $n-1$ edges of $MST(S)$ contributes an index to J. ∎

In the next section, we discuss several constructions of collective bottleneck spanners $\mathcal{G}$ of S.

5 Examples of collective bottleneck spanners

5.1 The Yao-graph

In this section, we consider the *Yao-graph* [16], which is also known as the *geographic neighborhood graph*. Let S be a set of n points in the plane, and let $0 < \theta < \pi/4$ 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 $\vec{p}$. (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.* [5] 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. In the following two lemmas, we will analyze the path constructed by this algorithm.

Lemma 5 *Let p, q, and r be as above. We have*

$$|rq| \leq |pq| - (\cos\theta - \sin\theta)|pr|.$$

Proof. Let α be the angle between the line segments pq and pr, and let r' be the orthogonal projection of r onto pq; see Figure 1. Observe that $\alpha \leq \theta$. We have $|rr'| = |pr|\sin\alpha \leq |pr|\sin\theta$ and $|pr'| = |pr|\cos\alpha \geq |pr|\cos\theta$. It follows that

$$|rq| \leq |rr'| + |r'q| = |rr'| + |pq| - |pr'| \leq |pr|\sin\theta + |pq| - |pr|\cos\theta,$$

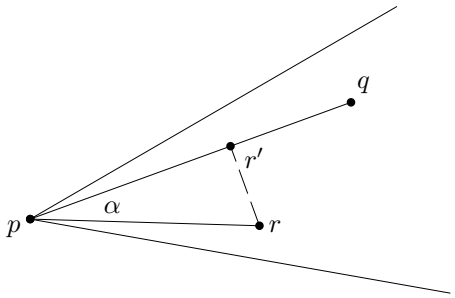

Figure 1: *Illustrating the proof of Lemma 5.*

---

proving the lemma. ∎

Lemma 6 *Let $t = 1/(\cos\theta - \sin\theta)$, and let p and q be any two points of S. There is a path between p and q in the Yao-graph $Y(S,\theta)$ whose length is less than or equal to $t|pq|$, and all of whose edges have length at most $|pq|$.*

Proof. The proof is by induction on the rank of $|pq|$ in the sequence of all $\binom{n}{2}$ distances determined by pairs of points in S. If p and q form a closest pair in S, then the lemma holds because p and q are connected by an edge in $Y(S,\theta)$.

We now assume that p and q do not form a closest pair. Furthermore, we assume that the lemma holds for all pairs of points whose distance is less than $|pq|$. Let C be the cone in $\mathcal{C}$ such that $q \in C_p$, and let (p,r) be the edge in $Y(S,\theta)$ with $r \in C_p$. If $r = q$, then (p,q) is an edge in $Y(S,\theta)$ and the lemma holds. So assume that $r \neq q$. Observe that $|pr| \leq |pq|$. Since $0 < \theta < \pi/4$, Lemma 5 implies that $|rq| < |pq|$. Therefore, by the induction hypothesis, there is a path P between r and q in $Y(S,\theta)$ whose length is less than or equal to $t|rq|$, and all of whose edges have length at most $|rq|$. Let P' be the path obtained by concatenating (p,r) and P. Then the length of each edge of P' is less than or equal to $|pq|$. By Lemma 5, we have

$$|P'| = |pr| + |P| \leq |pr| + t|rq| \leq |pr| + t|pq| - |pr| = t|pq|.$$

This completes the proof. ∎

Let $\epsilon > 0$ be a real constant. We choose a constant $0 < \theta < \pi/4$ such that $1/(\cos\theta - \sin\theta) \leq 1 + \epsilon$ and consider the graph $Y(S,\theta)$. Let

m denote the number of edges in this graph. Then $m \leq (2\pi/\theta)n = O(n)$. 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 $1 \leq k \leq m$, we denote by $Y^{(j_k)}$ the graph with vertex set S 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 i with $1 \leq i \leq \binom{n}{2}$. We may assume that p and q are connected by a path in the graph $K^{(i)}$. 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

$$\delta \leq (1+\epsilon) \cdot \delta^{(i)}(p,q). \tag{1}$$

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 $|xy| \leq L_i$. By Lemma 6, there is a path P_{xy} between x and y in the graph $Y(S,\theta)$ whose length is at most $(1+\epsilon)|xy|$ and all of whose edges are of length at most $|xy|$. Since P_{xy} is a path in $Y(S,\theta)$, each of its edges has in fact length at most L_{j_k}. That is, 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 (1). We have shown the following theorem.

Theorem 7 *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. Each graph in this collection has $O(n)$ edges.*

If the set S consists of the center p of a circle and $n-1$ points on the boundary of this circle, then the degree of p in $Y(S,\theta)$ is $n-1$. Therefore, the bottleneck graphs in the collection $\mathcal{G}$ are not of bounded degree.

Arya *et al.* [1] have shown how to combine the Yao-graph with so-called *single-sink spanners* to obtain a $(1+\epsilon)$-spanner of bounded degree for any point set S. In the same way as in Lemma 6, it can be shown that this spanner has the property that any two points p and q of S are connected by a path whose length is less than or equal to $(1+\epsilon)|pq|$, and all of whose edges have length at most $|pq|$. This implies the following result.

Theorem 8 *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. The maximum degree of each graph in this collection is bounded by a constant.*

Even though the bottleneck graphs in Theorems 7 and 8 are small in size, they are difficult to preprocess for shortest path queries. In the next section, we will see how to obtain a collective bottleneck spanner consisting of $O(n)$ planar graphs. As we will show in Section 6, planar graphs have the advantage that shortest path queries can be answered efficiently.

5.2 The Delaunay triangulation

Let S be a set of n points in the plane. We assume for simplicity that these points are in general position, i.e., no three points of S are collinear, and no four points of S are cocircular. Let $DT(S)$ be the Delaunay triangulation of S, see [3]. We will show that for any two points p and q of S, there exists a path P between p and q in $DT(S)$ such that (i) the length $|P|$ of P is less than or equal to $2\pi/(3\cos(\pi/6)) \cdot |pq|$, and (ii) no edge on P has length more than $|pq|$. 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 [10]. (The proof in [10] may produce a path between p and q that contains an edge whose length is larger than $|pq|$.)

We start by stating the main lemma that is needed for our proof. We remark that this lemma is similar to Lemma 1 in [10].

Before we can state the main lemma, we have to define the notion of upper angle. Let p and q be two distinct points of S, and let L be the line through p and q. Assume that L is not vertical and that p is to the left of q. Let L_+ and L_- be the open halfplanes consisting of all points that are above and below L, respectively. Let C be any circle that has p and q on its perimeter, and let m be the center of C.

Assume that no point of S is in the intersection of L_- and the interior of C. Then the *upper angle* of p and q is defined as the angle by which we have to rotate the line segment mp in clockwise order so that it coincides with mq. See Figure 2 for an illustration.

Now assume that no point of S is in the intersection of L_+ and the interior of C. Then the *upper angle* of p and q is defined as the angle by which we have

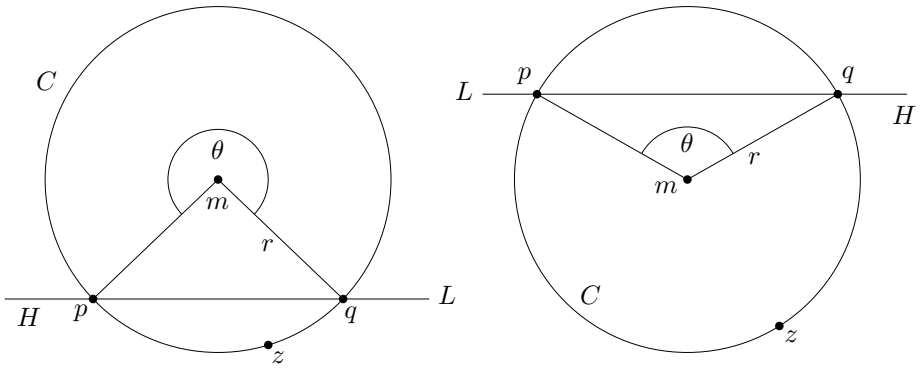

Figure 2: *Illustrating the assumptions in Lemma 9. There are two cases depending on whether* $0 < \theta \leq \pi$ *or* $\pi < \theta < 2\pi$. *There are no points of* S *in the part of the interior of* C *that is below the line* L.

---

to rotate the line segment mp in counterclockwise order so that it coincides with mq.

Lemma 9 *Let* p *and* q *be two distinct points of* S, *and let* L *be the line through* p *and* q. *Let* C *be any circle that has* p *and* q *on its perimeter, and let* r *and* m *be the radius and center of* C, *respectively. Assume that no point of* S *is in the intersection of* L_- *and the interior of* C, *or no point of* S *is in the intersection of* L_+ *and the interior of* C. *Let* θ *be the upper angle of* p *and* q. *Then there exists a path* P *in* $DT(S)$ *between* p *and* q *such that*

1. *the length of* P *is less than or equal to* $r\theta$, *and*
2. *the length of each edge on* P *is less than or equal to* $|pq|$.

5.2.1 Proof of Lemma 9

We will assume, for ease of presentation, that p and q are both on the x-axis, p is to the left of q, and no point of S is in the part of the interior of C that is below the x-axis.

If (p, q) is an edge of the convex hull of S, then it is also an edge of $DT(S)$. In this case, Lemma 9 clearly holds. So we assume from now on that (p, q) is not a convex hull edge.

Our goal is to prove Lemma 9 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.

To normalize C, we move the center m of C downwards along the bisector of p and q; during this movement, we change C so that it always contains the points p and q. We stop this process at the moment when the part of C below the x-axis hits a point, say z, of S. Observe that z exists, because otherwise (p, q) would be an edge of the convex hull of S. Observe that during the movement, the radius r and the upper angle θ of C change. It is easy to see, however, that the product $r\theta$ decreases. Hence, it suffices to prove Lemma 9 for the new circle C, which we will refer to as a *normalized* circle.

Hence, from now on, C is a normalized circle through the points p, q, and z, where z is below the x-axis. This circle has center m and radius r. There are no points of S in the part of the interior of C that is below the x-axis.

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 $O(n^2)$ normalized circles. We proceed by induction on the rank of the upper angles θ of these circles.

For the base case, assume that the angle θ of C is minimum over all normalized circles. We claim that (p, q) is an edge of $DT(S)$, which will prove that Lemma 9 holds for the points p and q. To prove the claim, assume that (p, q) is not an edge of $DT(S)$. Then, by the definition of the Delaunay triangulation, the part of the interior of C that is above the x-axis contains at least one point of S.

For any point t of S that is in the interior of C, let D_t be the circle through p, q, and t. Choose the point t such that no point of S lies in the part of the interior of D_t that is above the x-axis. (Observe that such a t must exist.) We will write D instead of D_t. By symmetry, we may assume that t is to the left of m; see Figure 3.

Let C_1 be the circle through p and t whose center lies on the segment pm. We denote the center of C_1 by m_1. By the definition of D, no point of S lies in the part of the interior of C_1 that is below the line through p and t. Let θ_1 be the upper angle $\angle pm_1t$. Observe that (i) the ray from m_1 in direction $\overrightarrow{mq}$ intersects the x-axis in a point of C_1, and (ii) t is above this ray. This implies that $\theta_1 < \theta$.

If C_1 is a normalized circle, then we have obtained a contradiction to our assumption that θ is the smallest angle. In general, however, C_1 will not be a normalized circle. Therefore, we proceed as follows. We move the center m_1

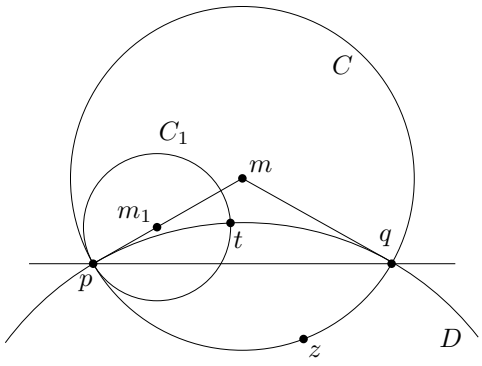

Figure 3: *Illustrating the base case. The upper angle $\angle pm_1t$ is equal to θ_1.*

---

of C_1 down along the bisector of p and t; during this movement, we change C_1 so that it always contains p and t. We stop at the moment when the part of C_1 that is below pt hits a point, say x, of S. (Observe that x exists: it is equal to z or a point of S that is hit upon earlier.) The new circle C_1' is a normalized circle, and its upper angle θ_1' is less than or equal to the upper angle θ_1 of the initial circle C_1. Hence, $\theta_1' < \theta$, which is a contradiction. This completes the base case.

For the inductive step, we again consider the normalized circle C with upper angle $\theta = \angle pmq$. We assume that θ is not the minimum angle. Furthermore, we assume that Lemma 9 holds for all normalized circles whose upper angles are less than θ. If (p, q) is an edge of $DT(S)$, then Lemma 9 holds for the circle C. So we may assume that (p, q) is not an edge of $DT(S)$. We define the point t and the circle D as in the base case. We may assume without loss of generality that t is to the left of m. Observe that $\theta \in (0, 2\pi)$. We will treat the cases $0 < \theta \leq \pi$ and $\pi < \theta < 2\pi$ separately. Before proceeding, we give some constructions that will be used in the sequel. (Refer to Figures 4 and 5.)

Let C_1 be the circle through p and t whose center lies on the segment pm. We denote the center and radius of C_1 by m_1 and r_1, respectively. Similarly, let C_2 be the circle through t and q whose center lies on the segment qm. We denote the center and radius of C_2 by m_2 and r_2, respectively. By our choice of D, no point of S lies in the part of the interior of C_1 that is below the line through p and t, and no point of S lies in the part of the interior of C_2 that is 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 m_1 and a_1 and the line through m_2 and a_2. We denote the center and radius of C_3 by m_3 and r_3, respectively.

We observe that the following four triangles are all similar isosceles triangles with two equal base angles, which we will denote by ϕ: $\triangle(p, m, q)$, $\triangle(p, m_1, a_1)$, $\triangle(a_2, m_2, q)$, and $\triangle(a_2, m_3, a_1)$.

As in the base case, both upper angles $\theta_1 := \angle p m_1 t$ and $\theta_2 := \angle t m_2 q$ are less than θ. We use the same construction as in the base case to move m_1 down along the bisector of p and t (changing C_1 such that it always contains p and t) until the part of C_1 that is below pt hits a point of S. Let C_1' be the resulting circle, and let θ_1' and r_1' be its upper angle and radius, respectively. Then C_1' is a normalized circle, $r_1'\theta_1' \leq r_1\theta_1$, and no point of S is in the part of the interior of C_1' that is below the line through p and t. Hence, by the induction hypothesis, there is a path P_1 between p and t in $DT(S)$, having length $|P_1| \leq r_1'\theta_1' \leq r_1\theta_1$, and all of whose edges have length at most $|pt|$. In a completely symmetric way, there is a path P_2 between t and q in $DT(S)$, having length $|P_2| \leq r_2\theta_2$, and all of whose edges have length at most $|tq|$.

Case 1: $0 < \theta \leq \pi$; see Figure 4.

Let P be the concatenation of P_1 and P_2. Then

$$|P| = |P_1| + |P_2| \leq r_1\theta_1 + r_2\theta_2 = r_1\theta + r_2\theta - (r_1(\theta - \theta_1) + r_2(\theta - \theta_2)) .$$

Since $0 < \theta \leq \pi$, a_2 is to the left of a_1. Observe that

1. the length of the upper arc of C_3 between a_2 and a_1 is equal to $r_3\theta$,
2. the length of the upper arc of C_1 between t and a_1 is equal to $r_1(\theta - \theta_1)$, and
3. the length of the upper arc of C_2 between a_2 and t is equal to $r_2(\theta - \theta_2)$.

Since the part of C_3 above the x-axis is convex and contained in both parts of the interiors of C_1 and C_2 that are above the x-axis, we have

$$r_1(\theta - \theta_1) + r_2(\theta - \theta_2) \geq r_3\theta.$$

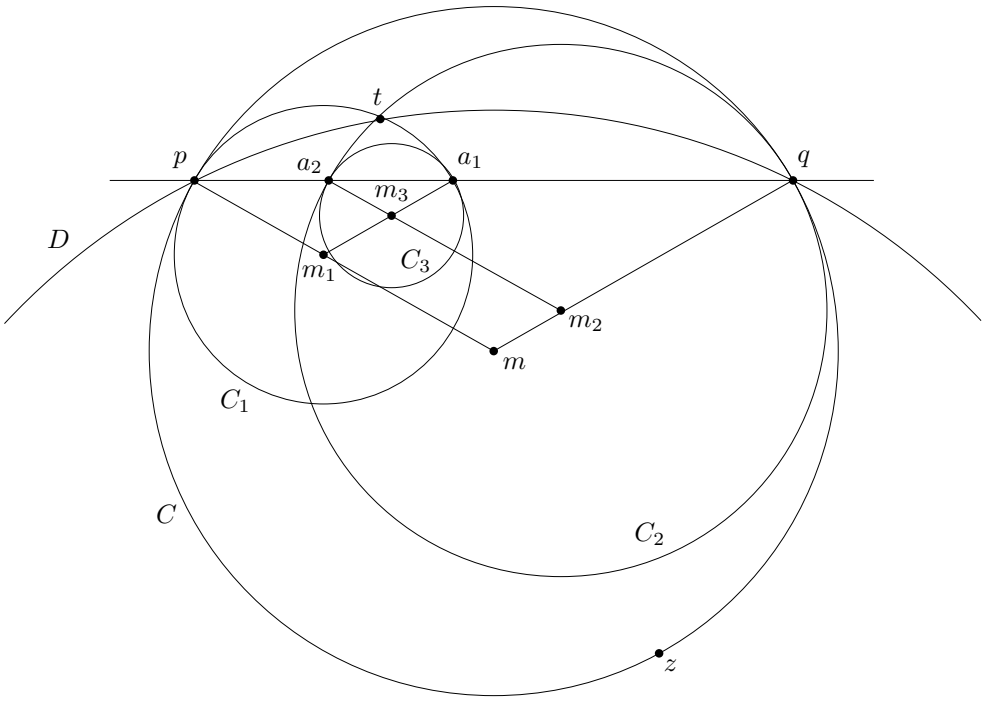

Figure 4: *Illustrating Case 1.*

Hence,

$$|P| \leq (r_1 + r_2 - r_3)\theta = \left(\frac{|pa_1|}{2\cos\phi} + \frac{|a_2q|}{2\cos\phi} - \frac{|a_2a_1|}{2\cos\phi}\right)\theta = \frac{|pq|}{2\cos\phi}\,\theta = r\theta.$$

Since $0 < \theta \leq \pi$, we have $|pt| \leq |pq|$ and $|tq| \leq |pq|$. Recall that all edges on P_1 have length at most $|pt|$ and all edges on P_2 have length at most $|tq|$. Therefore, the length of each edge on P is less than or equal to $|pq|$.

Case 2.a: $\pi < \theta < 2\pi$ and t is inside the circle having p and q as diameter; see Figure 5.

Let P be the concatenation of P_1 and P_2. Recall that $\theta_1 < \theta$ and $\theta_2 < \theta$. We have

$$|P| = |P_1| + |P_2| \leq r_1\theta_1 + r_2\theta_2 \leq (r_1 + r_2)\theta.$$

If a_2 is to the right of a_1, then

$$|P| \leq (r_1 + r_2)\theta = \left(\frac{|pa_1|}{2\cos\phi} + \frac{|a_2q|}{2\cos\phi}\right)\theta \leq \frac{|pq|}{2\cos\phi}\theta = r\theta.$$

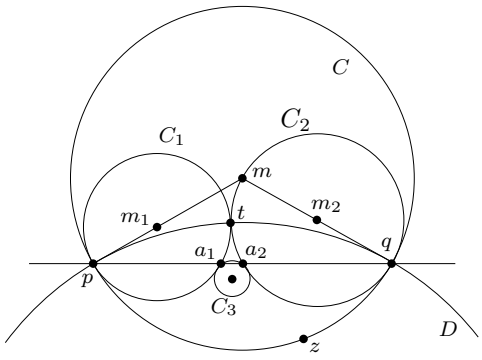

Figure 5: *Illustrating Case 2.a. The circle C_3 has m_3 as its center.*

If a_2 is to the left of a_1, then $|P| \leq r\theta$ by the same argument as in Case 1.

Since t is contained in the circle with p and q as diameter, we have $|pt| \leq |pq|$ and $|tq| \leq |pq|$. As a result, the length of each edge on P is less than or equal to $|pq|$.

Case 2.b: $\pi < \theta < 2\pi$ and t is outside the circle R with p and q as diameter; see Figure 6.

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 m_4 and r_4, respectively. Let θ_4 be the upper angle $\angle pm_4z$. (Recall our definition of upper angle given just before Lemma 9.) 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 m_5 and r_5, respectively. Let θ_5 be the upper angle $\angle zm_5q$.

Observe that no point of S is contained in the part of the interior of R that is above the x-axis. Therefore, there is no point of S in the part of the interior of C_4 that is above the line through p and z. Similarly, there is no point of S in the part of the interior of C_5 that is above the line through q and z. We also observe that both θ_4 and θ_5 are less than π and, hence, less than θ. After normalizing C_4 and C_5, in the same way as we did before, we can apply the induction hypothesis. Hence, there exists a path P_4 between p and z in $DT(S)$, having length $|P_4| \leq r_4\theta_4$, and all of whose edges have length at most pz. Similarly, there exists a path P_5 between z and q in $DT(S)$, having length $|P_5| \leq r_5\theta_5$, and all of whose edges have length at most zq. Let P be

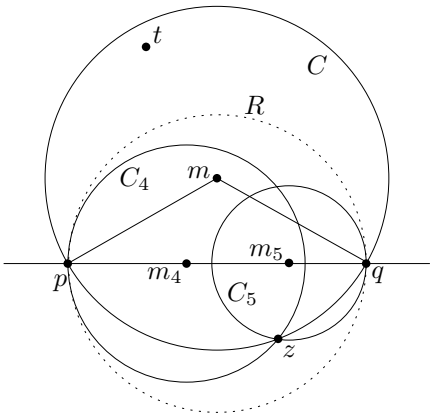

Figure 6: *Illustrating Case 2.b. The upper angle $\angle pm_4z$ is equal to θ_4; the upper angle $\angle zm_5q$ is equal to θ_5.*

---

the concatenation of P_4 and P_5. Then

$$|P| = |P_4| + |P_5| \leq r_4\theta_4 + r_5\theta_5 \leq (r_4 + r_5)\theta.$$

Since z is contained in R, both r_4 and r_5 are less than or equal to $|pq|/2$. Therefore, we have $|P| \leq r\theta$.

Finally, since both $|pz|$ and $|zq|$ are less than $|pq|$ (this again follows from the fact that z is contained in R), the length of each edge on P is less than or equal to $|pq|$. This concludes the proof of Lemma 9.

5.2.2 A collective bottleneck planar spanner

Using Lemma 9, we obtain the theorem below. We omit the proof, because it is basically the same as the proof of Theorem 1 in [10]. Replacing Lemma 1 in [10] by Lemma 9 guarantees that the length of each edge on the path is less than or equal to $|pq|$.

Theorem 10 *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 $|pq|$.*

We proceed as in Section 5.1. 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 $DT(S)$. Since $DT(S)$ is a planar graph, we have $m \leq 3n - 6$. 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}. As in Section 5.1, we obtain the following result.

Corollary 11 *Let S be a set of n points in the plane. The collection $\mathcal{G} := \{DT^{(j_k)} : 1 \leq k \leq m\}$ of planar graphs constitutes a collective bottleneck t-spanner of S, for $t = 2\pi/(3\cos(\pi/6))$. The number of graphs in this collection is less than or equal to $3n - 6$.*

6 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.

Using the results of Section 2, existence queries can be answered in $O(1)$ time. We will present a data structure of size $O(n^{5/2})$ that allows the shortest path whose edges have lengths at most L to be reported in $O(\sqrt{n} + \ell)$ time, where ℓ is the number of edges on the reported path.

Our solution uses the following result, due to Djidjev [7], for answering general shortest path queries in the entire planar graph G.

Lemma 12 ([7]) *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.*

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, we build the shortest path data structure of Lemma 12 for each of the graphs $G^{(i)}$. 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 $G^{(i)}$. The following observation is obvious.

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 $|e_i| \leq L < |e_{i+1}|$.*

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 $G^{(i)}$.*

2. *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. Using the data structure of Lemma 12, this takes $O(\sqrt{n} + \ell)$ time, where ℓ is the number of edges in the reported path.

The binary search tree T has size $O(n)$. Since G is planar, the number m of its edges is less than or equal to $3n-6$. Hence, 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 $O(n^{5/2})$. Thus, we obtain the following theorem.

Theorem 13 *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 $O(1)$ 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 10 and 13, then we obtain the following result.

Theorem 14 *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.*

7 Conclusion

We have presented efficient algorithms to solve a variety of geometric bottleneck problems. In each case, we show how to preprocess the data so that (approximate or exact) shortest path queries can be answered efficiently. In solving these problems, we use an array of tools such as minimum spanning trees, spanners, and the Delaunay triangulation.

The amount of preprocessing and space used in Theorems 3, 13, and 14, are very high. It is an open problem whether these bounds can be improved.

The graphs in the collective bottleneck spanner of Theorem 8 are of bounded degree, but they are not planar. On the other hand, the graphs in the collective bottleneck spanner of Corollary 11 are planar, but their degree may be unbounded. We leave it as an open problem to decide whether there exists a collective bottleneck spanner consisting of planar graphs, all having bounded degree. Observe that Bose *et al.* [4] have shown that a planar spanner of bounded degree can be computed for any point set. This spanner, however, does not have the property that each edge on a spanner path between two points p and q has length at most $|pq|$. Therefore, it is not clear if this result can be used in our context.

Acknowledgement

We thank an anonymous referee for bringing Neto's PhD thesis [12] to our attention.

References

[1] S. Arya, G. Das, D. M. Mount, J. S. Salowe, and M. Smid. Euclidean spanners: short, thin, and lanky. In *Proceedings of the 27th ACM Symposium on the Theory of Computing*, pages 489–498, 1995.

[2] M. A. Bender and M. Farach-Colton. The LCA problem revisited. In *Proceedings of the 4th Latin American Symposium on Theoretical Informatics*, volume 1776 of *Lecture Notes in Computer Science*, pages 88–94, Berlin, 2000. Springer-Verlag.

[3] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin, 2nd edition, 2000.

[4] P. Bose, J. Gudmundsson, and M. Smid. Constructing plane spanners of bounded degree and low weight. In *Proceedings of the 10th European Symposium on Algorithms*, volume 2461 of *Lecture Notes in Computer Science*, pages 234–246, Berlin, 2002. Springer-Verlag.

[5] 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.

[6] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 2nd edition, 2001.

[7] H. N. Djidjev. Efficient algorithms for shortest path queries in planar digraphs. In *Proceedings of the 22nd Workshop on Graph Theoretic Concepts in Computer Science*, volume 1197 of *Lecture Notes in Computer Science*, pages 151–165, Berlin, 1996. Springer-Verlag.

[8] D. Eppstein. Spanning trees and spanners. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 425–461. Elsevier Science, Amsterdam, 2000.

[9] D. Harel and R. E. Tarjan. Fast algorithms for finding nearest common ancestors. *SIAM Journal on Computing*, 13:338–355, 1984.

[10] J. M. Keil and C. A. Gutwin. Classes of graphs which approximate the complete Euclidean graph. *Discrete & Computational Geometry*, 7:13–28, 1992.

[11] G. Narasimhan and M. Smid. Approximation algorithms for the bottleneck stretch factor problem. *Nordic Journal of Computing*, 9:13–31, 2002.

[12] D. M. Neto. *Efficient Cluster Compensation for Lin-Kernighan Heuristics*. Ph.D. thesis, Department of Computer Science, University of Toronto, Toronto, Canada, 1999.

[13] B. Schieber and U. Vishkin. On finding lowest common ancestors: simplifications and parallelisations. *SIAM Journal on Computing*, 17:327–334, 1988.

[14] M. Smid. Closest-point problems in computational geometry. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 877–935. Elsevier Science, Amsterdam, 2000.

[15] I. Stojmenovic, editor. *Handbook of Wireless Networks and Mobile Computing*. Wiley and Sons, 2002.

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