Lower Bounds for Computing Geometric Spanners and Approximate Shortest Paths

Danny Z. Chen* Gautam Das† Michiel Smid‡

Abstract

We consider the problems of constructing geometric spanners, possibly containing Steiner points, for a set of n input points in d-dimensional space $\mathbb{R}^d$, and constructing spanners and approximate shortest paths among a collection of polygonal obstacles on the plane. The complexities of these problems are shown to be $\Omega(n \log n)$ in the algebraic computation tree model. Since $O(n \log n)$-time algorithms are known for solving these problems, our lower bounds are tight up to a constant factor.

Keywords: Computational geometry, spanner graphs, shortest paths, lower bounds.

1 Introduction

Geometric spanners are data structures that approximate the complete graph on a set of points in d-dimensional space $\mathbb{R}^d$, in the sense that the shortest path (based on such a spanner) between any pair of given points is not more than a factor of t longer than the distance between the two points in $\mathbb{R}^d$. Computing such approximate shortest paths is a fundamental topic in computational geometry because shortest path problems appear in many

*Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556, USA. E-mail: dchen@euclid.cse.nd.edu. The research of this author was supported in part by the National Science Foundation under Grant CCR-9623585.

†Math Sciences Dept., The University of Memphis, Memphis, TN 38152, USA. Supported in part by NSF Grant CCR-9306822. E-mail: dasg@msci.memphis.edu.

‡Department of Computer Science, University of Magdeburg, D-39106 Magdeburg, Germany. E-mail: michiel@isg.cs.uni-magdeburg.de.

application areas, such as robotics and VLSI design, and play vital roles in solving various geometric optimization problems.

Throughout this paper, we measure distances between points in d-dimensional space $\mathbb{R}^d$ with the Euclidean metric, where $d \geq 1$ is an integer constant. Let S be a set of n points in $\mathbb{R}^d$. We consider graphs $G = (V, E)$ such that (i) V is a set of points in $\mathbb{R}^d$, (ii) $S \subseteq V$, and (iii) the edges of G are straight-line segments in $\mathbb{R}^d$ that connect pairs of points in V. The *length* of an edge in G is defined as the distance between its endpoints. In such a graph, the *length* of a path is defined as the sum of the lengths of the edges on the path.

Let $t > 1$ be any real number. Consider a graph $G = (V, E)$ that satisfies (i), (ii), and (iii), such that for every pair p, q of points of S, there is a path in G between p and q of length at most t times the distance between p and q in $\mathbb{R}^d$. If $V = S$, then G is called a *t-spanner* for S. Otherwise, G contains additional vertices other than those in S, and we call G a *Steiner t-spanner* for S, and the points of $V \setminus S$ the *Steiner points* of G.

Several algorithms are known that for any fixed constant $t > 1$ and any set S of n points in $\mathbb{R}^d$, construct in $O(n \log n)$ time a t-spanner for S (i.e., without Steiner points) which consists of $O(n)$ edges. Note that the constant factors in the Big-Oh bounds of these algorithms depend on t and d. (See [3, 14, 15].) All these algorithms can be implemented in the algebraic computation tree model [2].

These algorithmic results naturally lead to the question of whether there are faster algorithms for constructing geometric spanners. In particular, if we allow a spanner to use significantly many Steiner points, is it possible to construct the spanner in $o(n \log n)$ time? In this paper, we give a negative answer to this question. We will prove that in the algebraic computation tree model, any algorithm that constructs a Steiner t-spanner for any set of n points in $\mathbb{R}^d$, has an $\Omega(n \log n)$ worst-case running time. This follows by a reduction from the *element uniqueness problem* [2, 12]. (See Section 2.) We also consider the spanner problem for the case when the input points are pairwise distinct. For such inputs, this reduction obviously does not work. In Section 2, we will prove that the $\Omega(n \log n)$ lower bound for constructing Steiner t-spanners still holds for inputs consisting of pairwise distinct points. This lower bound is proved by using Ben-Or's theorem [2]. Note that this theorem cannot be applied directly, because it does not assume any restriction on the input. We will show, however, how to circumvent this.

The $O(n \log n)$-time algorithms for constructing t-spanners that were mentioned above all assume that t is a fixed constant. Our lower bound implies that these algorithms are optimal. In fact, our lower bound result says more: Even if t is part of the input, it still takes $\Omega(n \log n)$ time to compute a Steiner t-spanner. In particular, the lower bound holds even if t is a (very large valued) function of n.

In the last part of the paper (Section 3), we consider the problem of computing Steiner t-spanners *among obstacles.* In this case, we are given a set S of planar points, a set of polygonal obstacles on the plane, and a real number $t > 1$. A (Steiner) t-spanner is defined as before, except that now the edges of the spanner do not intersect the interior of any obstacle. There are several $O(n \log n)$-time algorithms for constructing such spanners, where n denotes the number of points of S plus the total number of obstacle vertices. (See [1, 4, 5, 6, 7].) We prove an $\Omega(n \log n)$ lower bound on the time complexity for solving this problem in the algebraic computation tree model. Note that although for certain cases of spanners this lower bound also follows from the results of Section 2, the proof techniques we use in Section 3 are different from those in Section 2. Furthermore, as we will also show, the proof given in Section 3 extends to the same lower bound for computing *approximate shortest paths among polygonal obstacles* on the plane and for computing other kind of spanners than those of Section 2. Again, there are $O(n \log n)$-time algorithms for the latter problem. (See [5, 6, 7, 8, 10, 11].) Hence, by our lower bound, these results are optimal.

2 The lower bound for constructing Steiner spanners

We assume that the reader is familiar with the algebraic computation tree model. (See Ben-Or [2] and Preparata and Shamos [12].) Throughout the rest of this section, we only consider algorithms that can be implemented in the algebraic computation tree model and that construct Steiner t-spanners with $o(n \log n)$ edges. (Clearly, any algorithm that constructs Steiner t-spanners with $\Omega(n \log n)$ edges takes $\Omega(n \log n)$ time.) Also, we will focus on algorithms that construct Steiner t-spanners for one-dimensional point sets. As will be seen, even the one-dimensional case has an $\Omega(n \log n)$ lower bound.

The *element uniqueness problem* is defined as follows: Given n real numbers $x_1, x_2, \ldots, x_n$, decide if they are pairwise distinct. It is well known that this problem has an $\Omega(n \log n)$ lower bound in the algebraic computation tree model. (See [2, 12].) We shall reduce this problem to that of constructing a Steiner t-spanner.

The main observation is that if $x_i = x_j$ for some i and j with $i \neq j$, then any Steiner t-spanner for $x_1, x_2, \ldots, x_n$ contains a path between x_i and x_j of length at most $t|x_i - x_j| = 0$. In particular, each edge on this path has length zero. Because the spanner may contain Steiner points, we have to be careful in formalizing this.

Let $\mathcal{A}$ be any algorithm that, given a sequence S of n real numbers $x_1, x_2, \ldots, x_n$ and a real number $t > 1$, constructs a Steiner t-spanner for S. We may assume that each vertex of the spanner graph constructed by $\mathcal{A}$ is labeled as either being an element of S or being a Steiner point.

We start with a preliminary reduction as follows. Let $S = (x_1, x_2, \ldots, x_n)$ be a sequence of n real numbers. Choose any real number $t > 1$. Using algorithm $\mathcal{A}$, construct a Steiner t-spanner G on the x_i's. Construct the subgraph G' of G such that G' contains the same vertices as G, and G' contains all edges of G of length zero. Compute the connected components of G'. For each component of G', check if it contains two or more distinct elements (i.e., elements having distinct indices) of S among its vertices. If this is the case for some component, output NO. Otherwise, output YES.

Hence, given the Steiner t-spanner G, we can solve the element uniqueness problem in a time proportional to the number of edges of G, which is $o(n \log n)$. Therefore, algorithm $\mathcal{A}$ has $\Omega(n \log n)$ running time.

This lower bound proof is unsatisfying in the sense that we often assume implicitly that all input elements are pairwise distinct. For such inputs, the above proof does not work. Therefore, in the rest of this section, we prove the following result.

Theorem 1 *Let $d \geq 1$ be an integer constant. In the algebraic computation tree model, any algorithm that, given a set S of n pairwise distinct points in $\mathbb{R}^d$ and a real number $t > 1$, constructs a Steiner t-spanner for S, takes $\Omega(n \log n)$ time in the worst case.*

As mentioned already, we prove this theorem for algorithms that compute Steiner t-spanners for one-dimensional point sets. Our proof makes use of the following well known result.

Theorem 2 (Ben-Or [2]) *Let W be any set in $\mathbb{R}^n$ and let $\mathcal{D}$ be any algorithm that belongs to the algebraic computation tree model and that accepts*

W. Let $\#W$ denote the number of connected components of W. Then the worst-case running time of $\mathcal{D}$ is $\Omega(\log \#W - n)$.

Throughout the rest of this section, $\mathcal{A}$ denotes any algorithm that, given a set S of n pairwise distinct real numbers and a real number $t > 1$, constructs a Steiner t-spanner for S with $o(n \log n)$ edges. (If any two distinct input elements of S are equal, then algorithm $\mathcal{A}$ is not defined.) Hence, the output of $\mathcal{A}$ is a graph having as its vertices the elements of S and (possibly) some additional Steiner points. Note that, although the elements of S are pairwise distinct, the output graph of $\mathcal{A}$ may have multiple vertices that represent the same numbers: There may be an element u of S and a Steiner point v that represent the same real number. Similarly, there may be Steiner points u and v that are different as vertices of the graph, but that represent the same real number. Hence, the graph may have edges of length zero. We assume that each vertex in the output graph of $\mathcal{A}$ is labeled as either being an element of S or being a Steiner point.

We will show that the worst-case running time of $\mathcal{A}$ is $\Omega(n \log n)$. This will prove Theorem 1.

To be able to apply Theorem 2, we need to define an appropriate algorithm $\mathcal{D}$ such that (i) $\mathcal{D}$ solves a decision problem, i.e., it outputs YES or NO, (ii) $\mathcal{D}$ has a running time that is within a constant factor of $\mathcal{A}$'s running time, and (iii) the set of YES-inputs of $\mathcal{D}$, considered as a subset of $\mathbb{R}^n$, consists of many ($\Omega(n!)$ in our case) connected components.

There is one problem here. We consider decision algorithms whose input consists of n real numbers that are *pairwise distinct*. The subset of $\mathbb{R}^n$ on which such an algorithm $\mathcal{X}$ is defined (i.e., the collection of sequences of n pairwise distinct real numbers) trivially has at least $n!$ connected components: Consider two distinct permutations π and ρ of $1, 2, \ldots, n$. Let i and j be indices such that $\pi(i) < \pi(j)$ and $\rho(i) > \rho(j)$. Any continuous curve in $\mathbb{R}^n$ between the points $P = (\pi(1), \pi(2), \ldots, \pi(n))$ and $R = (\rho(1), \rho(2), \ldots, \rho(n))$ contains a point Q whose i-th and j-th coordinates are equal. Algorithm $\mathcal{X}$ is not defined for the input that consists of the point Q. Therefore, P and R belong to different connected components of the set of valid inputs for $\mathcal{X}$. The problem is that we cannot apply Theorem 2 to algorithm $\mathcal{X}$. For example, $\mathcal{X}$ could be the algorithm that takes as input any sequence of n pairwise distinct real numbers, and simply outputs YES. (Again, if two input elements are equal, then algorithm $\mathcal{X}$ is not defined.) The subset of $\mathbb{R}^n$ accepted by this algorithm has at least $n!$ connected components, although its running time is bounded by a constant.

Therefore, in order to apply Theorem 2, we must carefully define algorithm $\mathcal{D}$, ensure that it can take *any* point of $\mathbb{R}^n$ as input, and that its set of YES-inputs still has $\Omega(n!)$ connected components. We will define algorithm $\mathcal{D}$ in three steps.

1. First, we define an algorithm $\mathcal{B}$ that takes pairwise distinct real numbers as input. This algorithm runs algorithm $\mathcal{A}$ on this input, and outputs the length ls of a shortest edge of non-zero length in the graph that $\mathcal{A}$ computes.

2. Next, we use algorithm $\mathcal{B}$ to define a positive real number ls^*. Algorithm $\mathcal{C}$ takes pairwise distinct real numbers as input. It runs algorithm $\mathcal{B}$ on this input, and outputs YES if and only if the output ls of $\mathcal{B}$ is greater than or equal to ls^*.

3. Finally, we change algorithm $\mathcal{C}$ such that it is well-defined on *any* input sequence of real numbers. The resulting algorithm is the algorithm $\mathcal{D}$ we are looking for.

In the following subsections, we will fill in the details.

2.1 Algorithm $\mathcal{B}$

Algorithm $\mathcal{B}$ does the following on an input consisting of n pairwise distinct real numbers $x_1, x_2, \ldots, x_n$ and a real number $t > 1$. It first runs algorithm $\mathcal{A}$ on the input $x_1, x_2, \ldots, x_n, t$. Let G be the Steiner t-spanner that is computed by $\mathcal{A}$. Considering all edges of G, algorithm $\mathcal{B}$ then selects a shortest edge of non-zero length, and outputs the length ls of this edge.

We introduce the following notation. For real numbers $x_1, x_2, \ldots, x_n$, we denote

$$mingap(x_1, x_2, \ldots, x_n) := \min\{|x_i - x_j| : 1 \leq i < j \leq n\}.$$

Lemma 1 *The shortest non-zero length ls that is output by algorithm $\mathcal{B}$ satisfies*

$$0 < ls \leq t \cdot mingap(x_1, x_2, \ldots, x_n).$$

Proof. Let i and j be two indices such that $|x_i - x_j| = \mathit{mingap}(x_1, x_2, \ldots, x_n)$. Note that since the input elements are pairwise distinct, we have $|x_i - x_j| > 0$. The graph G constructed by algorithm $\mathcal{A}$ must contain a path between x_i and x_j of length at most $t|x_i - x_j|$. Each edge on this path obviously has a length of at most $t|x_i - x_j|$. Moreover, this path contains at least one edge of non-zero length. ∎

Let $T_{\mathcal{A}}(n,t)$ and $T_{\mathcal{B}}(n,t)$ denote the worst-case running times of algorithms $\mathcal{A}$ and $\mathcal{B}$, respectively. Then the fact that the graph G has $o(n \log n)$ edges implies that $T_{\mathcal{B}}(n,t) \leq T_{\mathcal{A}}(n,t) + o(n \log n)$.

2.2 Algorithm $\mathcal{C}$

We fix an integer n, and a real number $t > 1$. For any permutation π of the n integers $1, 2, \ldots, n$, let ls_π be the output of algorithm $\mathcal{B}$ when given as input $\pi(1), \pi(2), \ldots, \pi(n), t$. Among all these $n!$ outputs, let ls^* be one that has the minimum value.

Now we can define algorithm $\mathcal{C}$. It only accepts inputs of our fixed length n, consisting of n pairwise distinct real numbers. On input $x_1, x_2, \ldots,$ x_n, algorithm $\mathcal{C}$ does the following. It first runs algorithm $\mathcal{B}$ on the input $x_1, x_2, \ldots, x_n, t$. Let ls be the output of $\mathcal{B}$. Algorithm $\mathcal{C}$ then outputs YES if $ls \geq ls^*$, and NO otherwise.

We remark that it is not necessary to compute ls^*, which would take a lot of time. For our proof, it is sufficient that algorithm $\mathcal{C}$ *exists*.

It is clear that the running time of algorithm $\mathcal{C}$ is within a constant factor of $\mathcal{B}$'s running time.

2.3 Algorithm $\mathcal{D}$

Algorithm $\mathcal{C}$ is defined only for inputs consisting of n pairwise distinct real numbers. As a result, $\mathcal{C}$ can safely perform operations such as $z := x/(x_i - x_j)$, for any real number x, without having to worry whether the denominator is zero or not. Our final algorithm $\mathcal{D}$ will take any point $(x_1, x_2, \ldots, x_n)$ of $\mathbb{R}^n$ as input. On input $x_1, x_2, \ldots, x_n$, $\mathcal{D}$ performs the same computation as $\mathcal{C}$ does on the same input, except that each operation of the form $z := x/y$ is performed by $\mathcal{D}$ as

if $y \neq 0$ **then** $z := x/y$ **else** output YES and terminate **endif**.

Since $\mathcal{C}$ is a well-defined algorithm, it will always be the case that $y \neq 0$ if the input consists of n pairwise distinct real numbers. When two input elements are equal, it may still be true that $y \neq 0$, although this is not necessarily the case.

It is clear that $\mathcal{C}$ and $\mathcal{D}$ give the same output when given as input the same sequence of n pairwise distinct real numbers. If these numbers are not pairwise distinct, then $\mathcal{C}$ is not defined, whereas $\mathcal{D}$ is, although its output may not have a meaning at all. Finally, note that the running time of $\mathcal{D}$ is within a constant factor of that of $\mathcal{C}$.

2.4 Analysis of algorithm $\mathcal{D}$

We will prove that the worst-case running time of algorithm $\mathcal{D}$ is $\Omega(n \log n)$. This will imply the same lower bound on the running time of our target algorithm $\mathcal{A}$.

Let W be the set of all points $(x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$ that are accepted by algorithm $\mathcal{D}$.

Lemma 2 *The set W has at least $n!$ connected components.*

Proof. Let π and ρ be two different permutations of $1, 2, \ldots, n$. We will show that the points

$$P := (\pi(1), \pi(2), \ldots, \pi(n))$$

and

$$R := (\rho(1), \rho(2), \ldots, \rho(n))$$

belong to different connected components of W. (Note that both these points are elements of W.) This will prove the lemma.

Since π and ρ are different permutations, there are indices i and j, $1 \leq i \leq n$, $1 \leq j \leq n$, such that $\pi(i) < \pi(j)$ and $\rho(i) > \rho(j)$.

Consider any continuous curve C in $\mathbb{R}^n$ that connects P and R. We will show that C contains a point Q that does not belong to the set W. From this, it will follow that P and R are in different connected components of W. Note that $Q = (q_1, q_2, \ldots, q_n)$ must have the property that $ls < ls^*$, where ls is the output of algorithm $\mathcal{B}$ on input $q_1, q_2, \ldots, q_n, t$. Moreover, in order to guarantee this, we have to take care that the coordinates of Q are pairwise distinct.

Since the curve C passes through the hyperplane $x_i = x_j$, it contains points for which the absolute difference between the i-th and j-th coordinates is positive but arbitrarily close to zero. However, for such points $Q = (q_1, q_2, \ldots, q_n)$, there may be two distinct indices k and ℓ such that $q_k = q_\ell$. We do not have any control over algorithm $\mathcal{D}$ when given such a point Q as input. Therefore, we proceed as follows. We take for Q the first point on the curve C such that

$$\mathit{mingap}(q_1, q_2, \ldots, q_n) \leq \mathit{ls}^*/(2t).$$

We will see below that the coordinates of Q are pairwise distinct. If we run algorithm $\mathcal{B}$ on input $q_1, q_2, \ldots, q_n, t$, then, by Lemma 1, the output ls satisfies $\mathit{ls} \leq t \cdot \mathit{ls}^*/(2t) < \mathit{ls}^*$. In the rest of the proof, we will formalize this.

Parametrize the curve C as $C(\tau)$, $0 \leq \tau \leq 1$, where $C(0) = P$ and $C(1) = R$. For each k, $1 \leq k \leq n$, we write the k-th coordinate of the point $C(\tau)$ as $C(\tau)_k$. Define

$$\tau_0 := \min\{0 \leq \tau \leq 1 : \mathit{mingap}(C(\tau)_1, C(\tau)_2, \ldots, C(\tau)_n) \leq \mathit{ls}^*/(2t)\}.$$

Note that τ_0 exists, because the curve C passes through the hyperplane $x_i = x_j$, and the function *mingap* is continuous along C.

Let $Q := C(\tau_0)$, and write this point as $Q = (q_1, q_2, \ldots, q_n)$. Then we have

$$\mathit{mingap}(q_1, q_2, \ldots, q_n) \leq \mathit{ls}^*/(2t).$$

Also, by Lemma 1, and since $C(0) = P \in W$, we have

$$\mathit{mingap}(C(0)_1, C(0)_2, \ldots, C(0)_n) \geq \mathit{ls}^*/t > \mathit{ls}^*/(2t).$$

The value of τ_0 is the first "time" at which the *mingap*-function is no bigger than $\mathit{ls}^*/(2t)$. Since this function is continuous along C, we have $\mathit{mingap}(q_1, q_2, \ldots, q_n) > 0$. Hence, $(q_1, q_2, \ldots, q_n)$ is a sequence of n pairwise distinct real numbers. Consider algorithm $\mathcal{D}$ when given this sequence as input. It runs algorithm $\mathcal{B}$ on the input $q_1, q_2, \ldots, q_n, t$. Let ls be the output of $\mathcal{B}$. By Lemma 1, we have

$$\mathit{ls} \leq t \cdot \mathit{mingap}(q_1, q_2, \ldots, q_n).$$

Hence, $\mathit{ls} \leq t \cdot \mathit{ls}^*/(2t) < \mathit{ls}^*$ and, therefore, algorithm $\mathcal{D}$ outputs NO. This implies that point Q does not belong to the set W. ■

Recall that we denote the number of connected components of the set W by $\#W$. Lemma 2 and Theorem 2 imply that any algorithm that accepts the set W has a running time

$$\Omega(\log \#W - n) = \Omega(n \log n).$$

Since $\mathcal{D}$ is one such algorithm, it follows that for our fixed values of n and t, the worst-case running time of $\mathcal{D}$ is at least equal to $c\, n \log n$, where c is a positive constant independent of n and t. This, in turn, implies that there is an input on which algorithm $\mathcal{A}$ takes time at least $c'n \log n$, for some constant $c' > 0$. Since c' does not depend on n and t, this implies that the lower bound holds for all values of n and t. This completes the proof of Theorem 1.

Our lower bound proof of Theorem 1 holds for input consisting of pairwise distinct points. In computational geometry, stronger assumptions on the input are often used, e.g., no two points are on an axes-parallel hyperplane, no three points are on a line, no four points are in a two-dimensional plane, etc. For such *general position* input, our lower bound proof does not hold; the proof only works for point sets in $\mathbb{R}^d$ that contain a sufficiently large collinear subset. We conjecture, however, that our proof technique can be extended to sets of points that are in general position.

3 Spanners and approximate shortest paths among obstacles on the plane

In this section, we consider lower bounds for the problems of computing approximate shortest paths and of constructing various spanners among disjoint polygonal obstacles on the plane with a total of n vertices. We prove that $\Omega(n \log n)$ is a lower bound on the time complexity for solving these problems in the algebraic computation tree model.

Let S be the set of polygonal obstacle vertices on the plane (isolated input points are considered as point-obstacles), and let $n = |S|$. Let $G = (V, E)$ be a graph such that (i) S is a subset of V, and (ii) the edges of G are straight-line segments on the plane that do not intersect the interior of any obstacle. Then the notion of spanners in the previous sections can be generalized such that G is a t-spanner for S if for any two obstacle vertices $u, v \in S$, there is a u-to-v path in G whose length is no more than t times the length of a shortest u-to-v obstacle-avoiding path on the plane. If $V = S$, then we call G

a t-spanner for S. Otherwise, G contains additional vertices (Steiner points), and we call G a Steiner t-spanner for S. Here $t > 1$ can be any real number, and can even depend on the input (e.g., as a function of n). If a spanner G is *planar*, then there is an embedding of the graph G on the plane, such that no two of its embedded edges properly cross each other (n.b., the edges need not be embedded as straight-line segments).

An obstacle-avoiding path connecting two points u and v on the plane is called a *t-short u-to-v path* if the length of that path is no more than t times the length of a shortest u-to-v obstacle-avoiding path on the plane.

We need to distinguish two kinds of spanners in this section: Explicitly represented spanners and implicitly represented spanners. The spanners considered in Section 2 are *explicitly represented spanners*, since there we assumed that each edge of such a spanner is specified or represented in some explicit manner. For example, the edges of such a spanner are to be output one by one, or are stored in a collection of adjacency lists, one list for each vertex of the spanner. Thus, constructing an explicitly represented spanner with n vertices and m edges requires $\Omega(n + m)$ time. Specifically, our lower bound results in Section 2 hold for explicitly represented spanners. Spanners in this section, however, are allowed to contain $\Omega(n \log n)$ edges, and if this is the case, the spanners are assumed to be representable in some implicit fashion, called *implicitly represented spanners.* That is, a certain representation of such a spanner (possibly with $\Omega(n \log n)$ edges) is assumed to be possible which takes only $o(n \log n)$ space to construct, such that information of the spanner can be obtained as if an explicit representation were used. For example, in $O(n)$ space, one could somehow represent a coloring of the points in S with several different colors, such that a spanner G of S contains only the edges whose endpoints are of different colors.

Our proof of the $\Omega(n \log n)$ lower bound for computing t-short obstacle-avoiding paths is inspired by the reduction that de Rezende, Lee, and Wu used to prove the $\Omega(n \log n)$ lower bound for computing rectilinear shortest obstacle-avoiding paths [13]. We reduce the problem of sorting an arbitrary set K of n pairwise distinct positive integers I_1, I_2, $\ldots$, I_n (whose range can be much larger than $O(n)$) to the t-short path and spanner problems we consider. This reduction is done mainly by constructing a geometric sorting device based on an (arbitrary) algorithm for the t-short path or spanner problem.

We first show that the problem of sorting n pairwise distinct (positive) integers has an $\Omega(n \log n)$ lower bound.

Lemma 3 *In the algebraic computation tree model, any algorithm that, given a set S of n pairwise distinct integers, sorts the elements of S takes $\Omega(n \log n)$ time in the worst case. Furthermore, the $\Omega(n \log n)$ lower bound also holds for the case of sorting n* positive *pairwise distinct integers.*

Proof. Yao showed in [16] that the element uniqueness problem for a sequence of n arbitrary integers has an $\Omega(n \log n)$ lower bound in the algebraic computation tree model. This implies the same lower bound for the problem of sorting a sequence of n arbitrary integers. We shall reduce the latter problem to that of sorting n pairwise distinct integers.

Let $(x_0, x_1, \ldots, x_{n-1})$ be a sequence of n arbitrary integers. For every $i = 0, 1, \ldots, n-1$, let $y_i := nx_i + i$. Then, $(y_0, y_1, \ldots, y_{n-1})$ so obtained is a sequence of n pairwise distinct integers. If π is the permutation such that $y_{\pi(0)} < y_{\pi(1)} < \cdots < y_{\pi(n-1)}$, then $x_{\pi(0)} \leq x_{\pi(1)} \leq \cdots \leq x_{\pi(n-1)}$ (i.e., sorting the y_i's immediately gives the x_i's in sorted order).

Reducing the problem of sorting n pairwise distinct integers to the case with only *positive* pairwise distinct integers is easy. One only needs to first find the minimum of the n given integers (in linear time) and then add to each such integer a sufficiently large positive integral value, in order to obtain a set of positive pairwise distinct integers to work with. ∎

Our lower bound proofs are based on the following framework of reduction (but the actual values of several parameters can vary from one proof to another). Consider a set K of n positive pairwise distinct integers I_1, I_2, $\ldots$, I_n. Let I_u (resp., I_v) be the smallest (resp., largest) integer in the set K (it is easy to find I_u and I_v in $O(n)$ time). For every integer $I_i \in K$, first map I_i to the point $p_i = (I_i, 0)$ on the plane, and then construct a rectangle R_i and a rectilinear notch N_i associated with p_i, as follows (see Figure 1). The edges of R_i and N_i are parallel to an axis of the coordinate system. The cutoff of the rectilinear notch N_i forms a $\delta \times \delta$ square s_i whose vertices are b, c, d, and e (the value of δ is carefully chosen to be sufficiently small and this will be done later). The point p_i is at the center of the square s_i and also at the center of the edge $\overline{gh}$ of R_i. The length of the edge $\overline{gh}$ is $\delta/2$, and the length of both the edges $\overline{ab}$ and $\overline{ef}$ of N_i is $\delta/4$. Let C be a large circle whose center is at the origin of the coordinate system and whose radius is dependent on the input value of t and on the specific problem (to be discussed later). We only consider the half of C to the right of the y-axis. Let the upper-right (resp., lower-right) corner of each R_i (resp., N_i) touch

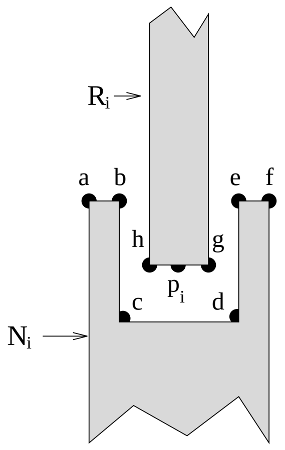

Figure 1: *The rectangle R_i and rectilinear notch N_i associated with the point p_i.*

the circle C (see Figure 2). Let the obstacle set consist of the R_i's and N_i's. It is not hard to observe that, because each R_i (resp., N_i) is contained in the circle C and its upper-right (resp., lower-right) corner touches C, and because C is the boundary of a convex region, the visibility graph G_V of the obstacle vertices in this geometric setting has only $O(n)$ edges (Figure 2). Moreover, observe that for a sufficiently small δ, the length of the shortest p_u-to-p_v obstacle-avoiding path in the visibility graph G_V among this set of obstacles is $< 2(I_v - I_u)$. Also, note that once the circle C is given, this reduction can be easily performed in $O(n)$ time.

We are now ready to prove the lower bounds of our problems.

Theorem 3 *In the algebraic computation tree model, any algorithm that, given a set of disjoint polygonal obstacles on the plane with a total of n vertices, two obstacle vertices p_u and p_v (of possibly certain point obstacles), and a real number $t > 1$, computes a t-short p_u-to-p_v obstacle-avoiding path based on G_V requires $\Omega(n \log n)$ time in the worst case.*

Proof. We reduce, as discussed above (Figure 2), the problem of sorting a set K of n positive pairwise distinct integers $I_1, I_2, \ldots, I_n$ to the problem of computing a t-short p_u-to-p_v obstacle-avoiding path based on G_V, where I_u (resp., I_v) is the smallest (resp., largest) integer in K. The key is to make the heights of the R_i's and N_i's very large, thus forcing any t-short p_u-to-p_v path in G_V to pass through some upper vertices of each N_i, in a sorted order

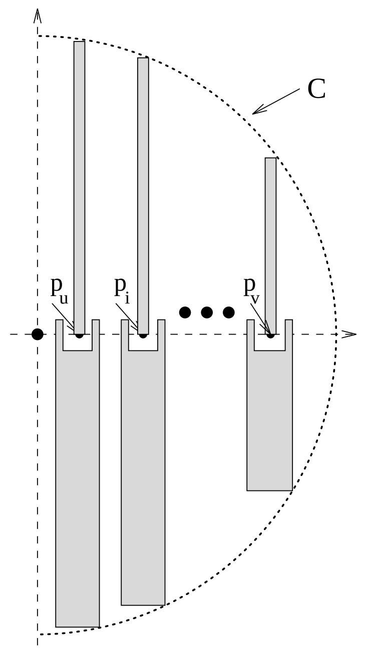

Figure 2: *Reducing integers I_1, I_2, ..., I_n to a geometric setting.*

of the p_i's; that is, we need to make C a sufficiently large circle. Specifically, we let δ be any real number with $0 < \delta < 1/8$, and let the height of N_v be $\geq \delta + 2t(I_v - I_u)$. (Note that this choice of δ's value ensures that the N_i's and R_i's are pairwise disjoint.) Next, we let C be the circle whose center is at the origin and that passes through the lower-right corner of N_v, and let other obstacles R_i and N_i touch C as discussed above (Figure 2). Now, observe that the height of any R_i (resp., N_i), for each $i = 1, 2, \ldots, n$, is equal to or larger than the height of N_v (which is $\geq \delta + 2t(I_v - I_u)$), because R_i (resp., N_i) touches the half circle of C on or to the left of N_v. Also, observe that, due to the heights of the obstacles in this setting (Figure 2), any t-short p_u-to-p_v path in G_V must pass through at least one of the vertices b and e of each N_i, and the length of such a t-short path is $< 2t(I_v - I_u)$. We assume without loss of generality that the reduction has associated the index i with the vertices b and e of each N_i.

After the $O(n)$ time reduction, we simply use an (arbitrary) algorithm to compute a t-short p_u-to-p_v path in this geometric setting. Then tracing this path from p_u to p_v (in $O(n)$ time) will give us a sorted sequence of the integers I_1, I_2, $\ldots$, I_n. Therefore, the $\Omega(n \log n)$ lower bound holds for the t-short path problem on an obstacle-scattered plane. ∎

It is worth pointing out that Theorem 3 can be easily generalized to obstacle-scattered spaces of higher dimensions.

Theorem 4 *In the algebraic computation tree model, any algorithm that, given a set of disjoint polygonal obstacles on the plane with a total of n vertices, and a real number $t > 1$, constructs a t-spanner of G_V (explicitly or implicitly represented) requires $\Omega(n \log n)$ time in the worst case.*

Proof. We first perform exactly the same reduction as in the proof of Theorem 3 (with the same values for the parameters). We then use an (arbitrary) algorithm to construct a t-spanner G of G_V such that the vertices of G are precisely the obstacle vertices (it does not matter whether G is explicitly or implicitly represented). Now observe that, because of the chosen heights of the obstacles R_i and N_i, G must contain a t-short p_u-to-p_v path P that does not pass through any upper (resp., lower) vertices of the R_i's (resp., N_i's). Furthermore, observe that G contains only $O(n)$ edges because the visibility graph G_V of the obstacle vertices in this setting has only $O(n)$ edges.

From the spanner G, we remove all its edges whose lengths are $\geq t(I_v - I_u)$ (this can be easily done in $O(n)$ time), and let the graph thus resulted be G'. Note that no edge on any t-short p_u-to-p_v path P in G is removed from G since the length of every such edge is $< I_v - I_u \leq t(I_v - I_u)$. More importantly, G' has the following property: There is no path in G' from p_u to any upper (resp., lower) vertex of the R_i's (resp., N_i's). If this were not the case, then there would be a path P' in G' from p_u to (say) an upper vertex of an R_i. Without loss of generality, let R_j be the rectangle such that its upper vertex z first appears in P'. But then the first edge on P' connecting with z cannot be adjacent to an upper vertex of another R_k, and, consequently, this edge is of a length $\geq t(I_v - I_u)$, a contradiction to the definition of G'.

It is now an easy matter to find in G' a p_u-to-p_v path P^* in $O(n)$ time (say, by performing a depth-first search in G'). Note that P^* need not pass through a particular point p_i. But, for each point p_i, P^* must pass through some of the vertices in $\{a, b, c, d, e, f, g, h\}$ that are associated with p_i (see Figure 1). We "color" all the vertices in $\{a, b, c, d, e, f, g, h\}$ associated with a point p_i by a "color" i. Note that, if we travel along the p_u-to-p_v path P^*, the vertices of the same "color" need not appear consecutively along P^*.

Nevertheless, we can obtain a sorted sequence of the input integers from P^*, as follows: We travel along P^* from p_u to p_v, and whenever we encounter a vertex on P^* with a "color" i for the first time, we report integer I_i. This traveling process can be easily done in $O(n)$ time. That the "colors" we encounter and output in this manner are in the sorted order of the input integers follows from the fact that P^* is a path of the visibility graph G_V that does not pass through the upper (resp., lower) vertices of the R_i's (resp., N_i's). This proves the theorem. ∎

Theorem 5 *In the algebraic computation tree model, any algorithm that, given a set of disjoint polygonal obstacles on the plane with a total of n vertices, and a real number $t > 1$, constructs an explicitly represented Steiner t-spanner that contains $o(n \log n)$ Steiner points and $o(n \log n)$ edges requires $\Omega(n \log n)$ time in the worst case.*

Proof. The proof of this theorem can be viewed as a generalization of the ideas used in proving Theorem 4. We use basically the same reduction framework as in the proof of Theorem 4 (i.e., reducing the problem of sorting positive pairwise distinct integers to the geometric setting as shown in Figure 2). However, due to some special properties of the Steiner t-spanners we consider, we need to choose carefully the values for certain parameters of the geometric setting, and to use several additional observations and ideas in this proof.

As in Theorem 3, we let δ be a real number with $0 < \delta < 1/8$. The value of the height of N_v (as well as the value of the radius of the circle C and the values of the heights of all other R_i's and N_i's) will be decided later.

Suppose that for a given set of height values of R_i's and N_i's, we have used an (arbitrary) algorithm to construct an explicitly represented Steiner t-spanner $G = (V, E)$ with $o(n \log n)$ Steiner points and $o(n \log n)$ edges. Then $|V| = o(n \log n)$ because V consists of n obstacle vertices and $o(n \log n)$ Steiner points. It should be pointed out that the $o(n \log n)$ Steiner points can be scattered all over the obstacle-free region of the plane in any possible fashion. For example, many of the $o(n \log n)$ Steiner points could be on a same path in the spanner G.

As in the proof of Theorem 4, the key idea is to obtain from G an obstacle-avoiding path P^* from p_u to p_v, such that (1) P^* does not pass through any upper (resp., lower) vertices of the R_i's (resp., N_i's), and (2) an appropriate subsequence of vertices along P^* that corresponds to the sorted sequence of the n input integers can be identified in $o(n \log n)$ time. But, with the presence of Steiner points, preventing such a p_u-to-p_v path P^* in G from going through the upper (resp., lower) vertices of the R_i's (resp., N_i's) and appropriately identifying a subsequence of vertices along P^* need to be done in a different way from that of the proof of Theorem 4.

We first discuss how to prevent a certain p_u-to-p_v obstacle-avoiding path from going through the upper (resp., lower) vertices of the R_i's (resp., N_i's). Observe that (1) there is a t-short p_u-to-p_v path P in G (because G is a t-spanner), and (2) the length of every edge on P is $< 2t(I_v - I_u)$ (since the shortest p_u-to-p_v obstacle-avoiding path on the plane is of a length $< 2(I_v - I_u)$, the length of P is $< 2t(I_v - I_u)$). Suppose that we have chosen to let the height of N_v be $> 2tn^2(I_v - I_u)$ (hence the radius of the circle C and the heights of all other R_i's and N_i's are decided accordingly). The reason for letting the height of N_v be $> 2tn^2(I_v - I_u)$ will soon be clear. We obtain another graph G' from G by removing from G all the edges whose lengths are $\geq 2t(I_v - I_u)$. Note that no edge on the path P is removed from G. More importantly, we claim that in G', there is no path from p_u to any upper (resp., lower) vertex of the R_i's (resp., N_i's). If this were not the case, then there would be a path P' in G' from p_u to (say) an upper vertex of an R_i. Without loss of generality, let R_j be the rectangle such that its upper vertex z first appears in P'. This means that when we travel along P' from p_u to z, we encounter no other upper (resp., lower) vertex of the R_i's (resp., N_i's) than z. There can be only $o(n \log n)$ vertices of G on the subpath of P' from p_u to z, and the length of this p_u-to-z path is $> 2tn^2(I_v - I_u)$ (by our choice of the height of N_v). It then follows that at least one edge on this p_u-to-z path is of a length $> 2t(I_v - I_u)$ (otherwise, the length of this p_u-to-z path in G' would be $\leq 2t(I_v - I_u) \times o(n \log n) < 2tn^2(I_v - I_u)$, a contradiction). But this is a contradiction to the definition of G'. (It is now clear that the factor of n^2 in our choice of the height of $N_v > 2tn^2(I_v - I_u)$ is for swamping the $o(n \log n)$ Steiner points of G'.) Note that because G has only $o(n \log n)$ edges, G' can be easily obtained in $o(n \log n)$ time.

We now discuss how to identify an appropriate subsequence of vertices along a p_u-to-p_v path P^* in G' that corresponds to the sorted sequence of the input integers. Note that due to the presence of Steiner points, a p_u-to-p_v path P^* in G' (which does not go through any upper (resp., lower) vertices of the R_i's (resp., N_i's)) need not pass through any vertex in the set $\{a, b, c, d, e, f, g, h\}$ associated with a point p_i (Figure 3). But, P^* does have to pass through the "alley" between every R_i and N_i, and this fact is

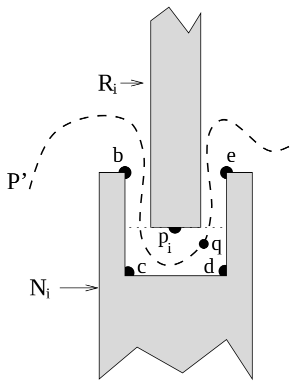

Figure 3: *Every p_u-to-p_v path P' in G' contains a vertex q of G in r_i.*

captured by the following observation:

($*$) For a point p_i, let r_i denote the half of the square s_i (recall that s_i is defined by the obstacle vertices b, c, d, and e associated with p_i) that is on or below the horizontal line passing through p_i (see Figure 3). Then every p_u-to-p_v path P' in G' goes through at least one point q in r_i such that q is either an obstacle vertex or a Steiner point of G.

We do the following traveling on such a p_u-to-p_v path P^* in G'.

1. Report integer I_u.

2. Suppose I_i is the previous input integer that has just been reported. If $I_i = I_v$, then stop.

3. Travel on path P^* to visit the next vertex v. Check whether v is in the rectangle r_{i+1}. If this is the case, then report integer I_{i+1}, let $i = i + 1$, and go to step 2; otherwise, repeat step 3.

Note that the vertex v reported in step 3 of the above procedure is the first vertex of P^* that is in the rectangle r_{i+1} (or, v is the first "color" $i + 1$ vertex of P^*). Hence our traveling process reports the input integers I_i in the sorted order (the correctness can be argued in a way similar to that of the proof of Theorem 4). The path P^* can be obtained in $o(n \log n)$ time by performing a depth-first search in G', and the traveling of P^* can also be done in the same time bound. This concludes the proof of this theorem. ■

Corollary 1 *In the algebraic computation tree model, any algorithm that, given a set of disjoint polygonal obstacles on the plane with a total of n vertices, and a real number $t > 1$, constructs an explicitly represented* planar *Steiner t-spanner with $o(n \log n)$ Steiner points requires $\Omega(n \log n)$ time in the worst case.*

Proof. Since such a planar Steiner t-spanner uses $o(n \log n)$ Steiner points, it contains only $o(n \log n)$ edges. Hence the corollary follows from Theorem 5. ∎

4 Concluding remarks

In this paper, we have proved $\Omega(n \log n)$ lower bounds for computing (Steiner) t-spanners and approximate shortest paths.

In [3, 14, 15], it is shown that for any set of n points in $\mathbb{R}^d$, a t-spanner can be constructed in $O(n \log n)$ time. Hence, the lower bound of Theorem 1 is tight.

Clarkson [7] gives an $O(n \log n)$ time algorithm for finding a $(1+\epsilon)$-short path among polygonal obstacles on the plane. Chew [4, 5, 6] gives $O(n \log n)$ time algorithms (based on planar 2-spanners) for computing 2-short paths among polygonal obstacles on the plane. More importantly, Hershberger and Suri [9] show that it is possible to compute an *exact* Euclidean shortest path among polygonal obstacles on the plane in $O(n \log n)$ time. Thus, the lower bound of Theorem 3 is tight.

Clarkson [7] and Das [8] show that $(1+\epsilon)$-spanners for shortest paths among polygonal obstacles on the plane can be built in $O(n \log n)$ time. Chew [4, 5, 6] constructs planar 2-spanners among polygonal obstacles on the plane in $O(n \log n)$ time. Therefore, the lower bound of Theorem 4 is also tight.

Finally, in [1], an $O(n \log n)$-time algorithm is given for constructing planar Steiner t-spanners among obstacles, with $O(n)$ Steiner points. Therefore, the lower bounds of Theorem 5 and Corollary 1, are tight.

We mentioned already at the end of Section 2 that our lower bound proof of Theorem 1 only works for point sets in $\mathbb{R}^d$ that contain a sufficiently large one-dimensional subset. Call a set S of points in $\mathbb{R}^d$ in *general position*, if for each k, $3 \leq k \leq d+1$, no k points of S are contained in a $(k-2)$-dimensional subspace of $\mathbb{R}^d$. We conclude this paper with the following conjecture.

Conjecture 1 *Let $d \geq 2$ be an integer constant. In the algebraic computation tree model, any algorithm that, given a set S of n points in $\mathbb{R}^d$ that are in general position, and a real number $t > 1$, constructs a Steiner t-spanner for S, takes $\Omega(n \log n)$ time in the worst case.*

Acknowledgements

The authors would like to thank Prof. Joseph O'Rourke for his comments that helped improve the presentation of the paper.

References

[1] S. Arikati, D.Z. Chen, L.P. Chew, G. Das, M. Smid, and C.D. Zaroliagis. *Planar spanners and approximate shortest path queries among obstacles in the plane.* Proceedings 4th Annual European Symposium on Algorithms (ESA), Lecture Notes in Computer Science, Vol. 1136, Springer-Verlag, Berlin, 1996, pp. 514-528.

[2] M. Ben-Or. *Lower bounds for algebraic computation trees.* Proceedings 15th Annual ACM Symposium on the Theory of Computing, 1983, pp. 80–86.

[3] P.B. Callahan and S.R. Kosaraju. *Faster algorithms for some geometric graph problems in higher dimensions.* Proceedings 4th Annual Symposium on Discrete Algorithms, 1993, pp. 291–300.

[4] L.P. Chew. *Constrained Delaunay triangulations.* Algorithmica **4** (1989), pp. 97–108.

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

[6] L.P. Chew. *Planar graphs and sparse graphs for efficient motion planning in the plane.* Computer Science Tech Report, PCS-TR90-146, Dartmouth College.

[7] K.L. Clarkson. *Approximation algorithms for shortest path motion planning.* Proceedings 19th Annual ACM Symposium on the Theory of Computing, 1987, pp. 56–65.

[8] G. Das. *The visibility graph contains a bounded-degree spanner.* Proceedings 9th Canadian Conference on Computational Geometry, 1997, pp. 70–75.

[9] J. Hershberger and S. Suri. *An optimal algorithm for Euclidean shortest paths in the plane.* SIAM Journal on Computing **28** (1999), pp. 2215–2256.

[10] J.S.B. Mitchell. *An optimal algorithm for shortest rectilinear paths among obstacles.* Proceedings 1st Canadian Conference on Computational Geometry, 1989, page 22.

[11] J.S.B. Mitchell. *L_1 shortest paths among polygonal obstacle in the plane.* Algorithmica **8** (1992), pp. 55–88.

[12] F.P. Preparata and M.I. Shamos. *Computational Geometry, an Introduction.* Springer-Verlag, New York, 1985.

[13] P.J. de Rezende, D.T. Lee, and Y.F. Wu. *Rectilinear shortest paths in the presence of rectangular barriers.* Discrete & Computational Geometry **4** (1989), pp. 41–53.

[14] J.S. Salowe. *Constructing multidimensional spanner graphs.* International Journal of Computational Geometry and Applications **1** (1991), pp. 99–107.

[15] P.M. Vaidya. *A sparse graph almost as good as the complete graph on points in K dimensions.* Discrete & Computational Geometry **6** (1991), pp. 369–381.

[16] A.C.-C. Yao. *Lower bounds for algebraic computation trees with integer inputs.* SIAM Journal on Computing **20** (1991), pp. 655–668.