Lower Bounds for Computing Geometric Spanners and Approximate Shortest Paths

Danny Z. Chen* Gautam Das† Michiel Smid‡

1 Introduction

Geometric spanners are data structures that approximate the complete graph on a set of points in the 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 points in $\mathbb{R}^d$.

Let τ be a constant such that $1 \leq \tau \leq \infty$. We measure distances between points in $\mathbb{R}^d$ with the L_τ-metric, where $d \geq 1$ is a 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 L_τ-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)–(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, if G contains additional vertices, we call G a *Steiner t-spanner* for S, and call 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. (See [3, 11, 12].) 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 in Section 2 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 lower bound even holds for inputs consisting of pairwise distinct points. 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, 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 in 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* in the plane and for computing other kind of spanners than those of Section 2. Again, there are $O(n \log n)$-time al-

*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. E-mail: dasg@next1.msci.memphis.edu. Supported in part by NSF Grant CCR-9306822.

‡Department of Computer Science, King's College London, Strand, London WC2R 2LS, United Kingdom. E-mail: michiel@dcs.kcl.ac.uk.

gorithms for the latter problem. (See [5, 6, 7, 8].) 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 [2, 9].) 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. 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.

We can reduce the *element uniqueness problem*—which has an $\Omega(n \log n)$ lower bound—to that of constructing a Steiner t-spanner. Hence, the latter problem has an $\Omega(n \log n)$ lower bound as well. However, this lower bound proof is unsatisfying in the sense that in computational geometry we often assume implicitly that all input elements are pairwise distinct. For such inputs, the above approach does not work. 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.*

Our proof makes use of the following result.

Theorem 2 (Ben-Or [2]) *Let W be any set in $\mathbb{R}^n$ and let $\mathcal{C}$ 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 $\mathcal{C}$ 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. 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, this graph 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 will show that the worst-case running time of $\mathcal{A}$ is $\Omega(n \log n)$. In order to apply Theorem 2, we have to define an appropriate algorithm $\mathcal{C}$ such that (i) $\mathcal{C}$ solves a decision problem, i.e., it outputs YES or NO, (ii) $\mathcal{C}$ has a running time that is within a constant factor of $\mathcal{A}$'s running time, and (iii) the set of YES-inputs of $\mathcal{C}$, considered as a subset of $\mathbb{R}^n$, consists of many (at least $n!$ in our case) connected components.

There is one problem here. We consider decision algorithms whose inputs consist of n real numbers that are *pairwise distinct*. The subset of $\mathbb{R}^n$ on which such an algorithm $\mathcal{X}$ is defined trivially has at least $n!$ connected components. We cannot apply Theorem 2 to algorithm $\mathcal{X}$. For example, $\mathcal{X}$ could be the algorithm that takes as input a sequence of n pairwise distinct real numbers, and simply outputs YES. The subset of $\mathbb{R}^n$ accepted by this algorithm has at least $n!$ connected components, although it has a running time of $O(1)$.

Therefore, to apply Theorem 2, we must carefully define algorithm $\mathcal{C}$. After we define algorithm $\mathcal{C}$ as specified above, we will further define a related algorithm $\mathcal{D}$ that takes any point of $\mathbb{R}^n$ as input, and whose set of YES-inputs still has at least $n!$ connected components. As the reader might expect, we start with defining an algorithm $\mathcal{B}$, before introducing algorithm $\mathcal{C}$.

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

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

Lemma 1 $0 < ls \leq t \cdot mg(x_1, x_2, \ldots, x_n)$.

We now fix an integer n and a real number $t > 1$. For any permutation π of the 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 minimal 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.

Since algorithm $\mathcal{C}$ only accepts inputs of our fixed length n, and since we also fixed t, we may assume that it "knows" the value ls^*. Algorithm $\mathcal{C}$ *exists*, although we have not explicitly computed ls^*.

Algorithm $\mathcal{C}$ is defined only for inputs consisting of n pairwise distinct real numbers. As a result, $\mathcal{C}$ can safely perform operations of the form $z := x/(x_i - x_j)$, 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 **fi**.

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.

We will prove that the 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 and Theorem 2 imply the lower bound on the running time of algorithm $\mathcal{D}$.

Lemma 2 $\#W \geq n!$.

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.

Let i and j, $1 \leq i, j \leq n$, be two indices 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. Since this curve 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 small. 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.

Parametrize the curve C as $C(\tau)$, $0 \leq \tau \leq 1$, where $C(0) = P$ and $C(1) = R$. For $1 \leq k \leq n$, we write the k-th coordinate of the point $C(\tau)$ as $C(\tau)_k$. Define τ_0 as the minimum value of τ, $0 \leq \tau \leq 1$, such that

$$mg(C(\tau)_1, C(\tau)_2, \ldots, C(\tau)_n) \leq ls^*/(2t).$$

Let $Q := C(\tau_0)$, and write this point as $Q = (q_1, q_2, \ldots, q_n)$. Then we have $mg(q_1, q_2, \ldots, q_n) \leq ls^*/(2t) < ls^*/t$. Also, $mg(C(0)_1, C(0)_2, \ldots, C(0)_n) \geq ls^* > ls^*/(2t)$. The value of τ_0 is the first "time" at which the mg-function is at most equal to $ls^*/(2t)$. Since this function is continuous along C, we have $mg(q_1, q_2, \ldots, q_n) > 0$. 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 $ls \leq t \cdot mg(q_1, q_2, \ldots, q_n)$. Hence, $ls < ls^*$ and, therefore, algorithm $\mathcal{D}$ outputs NO. This implies that point Q does not belong to the set W. ∎

3 Spanners and approximate shortest paths among obstacles in the plane

Let S be the set of obstacle vertices (isolated 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 in 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 in the plane. If $V = S$, then we call G a t-spanner for S. Otherwise, if G contains additional vertices (Steiner points), then we call G a Steiner t-spanner for S. If a spanner G is *planar*, then there is an embedding of the graph G in the plane, such that no two of its embedded edges properly cross each other.

An obstacle-avoiding path connecting two points u and v in 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 in 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. 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 with $o(n \log n)$ edges. Spanners in this section, however, are allowed to contain $\Omega(n \log n)$ edges, and if this is the case, the spanners, called *implicitly represented spanners*, are assumed to be representable in some implicit fashion. For example, one could, in $O(n)$ space, somehow represent a coloring of the points in S with several different colors, such that a spanner G of

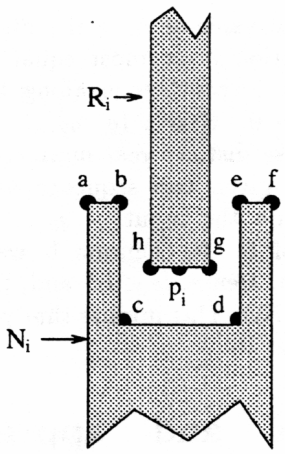

Figure 1: The rectangle R_i and rectilinear notch N_i associated with the point p_i.

S would contain 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 [10]. We reduce the problem of sorting a set K of n distinct positive integers I_1, I_2, ..., I_n to the t-short path and spanner problems we consider. This sorting problem has an $\Omega(n \log n)$ lower bound. The reduction is done mainly by constructing a geometric sorting device based on an (arbitrary) algorithm for the t-short path or spanner problem.

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, ..., I_n. Let I_u (resp., I_v) be the smallest (resp., largest) integer in the set K. For every integer $I_i \in K$, first map I_i to the point $p_i = (I_i, 0)$ in 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. 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 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, the visibility graph of the obstacle vertices in this geometric setting has only $O(n)$ edges (Figure 2). Moreover, observe that the length of the shortest p_u-to-p_v obstacle-avoiding path 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.

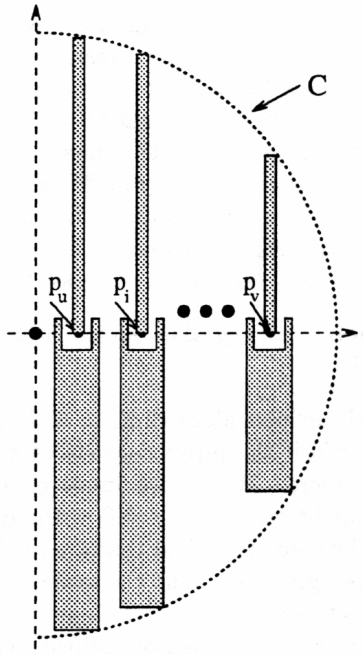

Figure 2: Reducing integers $I_1, I_2, \ldots, I_n$ to a geometric setting.

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

Proof: We reduce the problem of sorting a set K of n positive pairwise distinct integers I_1, I_2, ..., I_n to the problem of computing a t-short p_u-to-p_v obstacle-avoiding path in the plane, 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 the (unique) t-short p_u-to-p_v path to go through the points p_i, in sorted order. Specifically, we let δ be any real number with $0 < \delta < 1/8$, and let the height of N_v be $\geq \delta + 2t(I_v - I_u)$. 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 no smaller than the height of N_v (which is $\geq \delta + 2t(I_v - I_u)$). Also, observe that there can be only one t-short p_u-to-p_v path in the plane. Furthermore, this t-short p_u-to-p_v path goes through the edge of each R_i that contains p_i, and the length of this t-short path is $< 2t(I_v - I_u)$. In fact, for every value t' with $1 \leq t' \leq t$, the t'-short p_u-to-p_v path in this geometric setting is identical to the t-short p_u-to-p_v path. 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 will give us a sorted sequence of the integers $I_1, I_2, \ldots, I_n$. ∎

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

Proof: We first perform 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 whose vertices are precisely the obstacle vertices. 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.

From the spanner G, we remove all its edges whose lengths are $\geq t(I_v - I_u)$, and let the graph so resulted be G'. Note that no edge on the t-short p_u-to-p_v path P is removed from G. More importantly, 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. W.l.o.g., let R_j be the rectangle such that its upper vertex z first appears in P'. But then the 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.

It is now an easy matter to find in G' a p_u-to-p_v path P^* in $O(n)$ time. 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 two times. In the first traveling along P^*, we keep track of, for each "color", the last vertex with that "color" that we encounter. This traveling process can be easily done in $O(n)$ time. After the first traveling along P^*, we travel along P^* again, and this time, we output along the order of P^* the "color" vertices that we have kept track of as the result of our first traveling on P^*. That the "colors" we 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 that does not pass through the upper (resp., lower) vertices of the R_i's (resp., N_i's). ∎

Theorem 5 *In the algebraic computation tree model, any algorithm that, given a set of disjoint polygonal obstacles in 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.*

Proof: We use basically the same reduction framework as in the proof of Theorem 4. However, we need to choose carefully the values for a few parameters of the geometric setting and to use several additional observations and ideas in this proof. In particular, we let δ be a positive number $\leq \min\{1/(2tn^2), 1/8\}$, and let the height of N_v be a value $\geq \delta + 2tn^2(I_v - I_u)$. Once the value of the height of N_v is decided, the value of the radius of the circle C and the values of the heights of all the other R_i's and N_i's can be decided accordingly.

Suppose that 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 $|V| = o(n \log n)$. The key idea is to obtain from the spanner 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) with an appropriate "coloring" of a subset of the vertices in V, the "colors" of the vertices along P^* can lead to finding the sorted sequence of the input integers. However, 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 coloring a subset of the vertices in V must 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, and (2) the length of every edge on P is $< 2t(I_v - I_u)$. 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

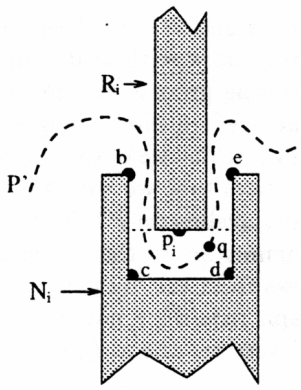

Figure 3: Every p_u-to-p_v path P' in G' contains a vertex q of G in r_i.

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. W.l.o.g, let R_j be the rectangle such that its upper vertex z first appears in P'. It 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 path along P' from p_u to z, and the length of this p_u-to-z path is $\geq 2tn^2(I_v - I_u)$. 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'.

We now discuss how to "color" a subset of the vertices in G'. Note that because of the presence of Steiner points, a p_u-to-p_v path P' in G' (which cannot 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), even though P' does have to pass through the "alley" between R_i and N_i. Our "coloring" method is based on the following observation:

(*) For a point p_i, let r_i be the portion 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. Furthermore, the distance between p_i and q is $\leq \delta$, and there is a t-short p_i-to-q path in G' whose length is $\leq t\delta$.

We do the "coloring" as follows. We first obtain from G' another graph G'', by removing from G' all the edges whose lengths are $\geq 1/n^2$. We then have the following claims on G''. (i) There is no path in G'' connecting two distinct points p_i and p_j; (ii) For every point p_i and every vertex w of G such that the length of the shortest p_i-to-w obstacle-avoiding path in the plane is $\leq \delta$, there is a path in G'' connecting p_i and w.

Note that the second claim implies that there is a path in G'' connecting p_i and the point q, where q is defined as in the observation (*) given above. Also, note that it is trivial to obtain G'' from G' in $o(n \log n)$ time.

Based on the above two claims, the rest of the "coloring" process is done as follows: For every point p_i, compute the connected component in G'' that contains p_i (by performing a depth-first search in G''), and "color" this connected component with a "color" i. This computation certainly takes $o(n \log n)$ time. The rest of the proof proceeds as in the proof of Theorem 4. ∎

Corollary 1 *In the algebraic computation tree model, any algorithm that, given a set of disjoint polygonal obstacles in 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.*

Corollary 1 implies that the $O(n \log n)$-time algorithms in [1] for constructing planar Steiner t-spanners with $O(n)$ Steiner points are optimal.

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.* Manuscript, 1996.

[2] M. Ben-Or. *Lower bounds for algebraic computation trees.* 15th STOC, 1983, pp. 80–86.

[3] P.B. Callahan and S.R. Kosaraju. *Faster algorithms for some geometric graph problems in higher dimensions.* 4th SODA, 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.* JCSS **39** (1989), pp. 205–219.

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

[7] K.L. Clarkson. *Approximation algorithms for shortest path motion planning.* 19th STOC, 1987, pp. 56–65.

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

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

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

[11] J.S. Salowe. *Constructing multidimensional spanner graphs.* IJCGA **1** (1991), pp. 99–107.

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