Constructing plane spanners of bounded degree and low weight

Prosenjit Bose* Joachim Gudmundsson† Michiel Smid*

Abstract

Given a set S of n points in the plane, we give an $O(n \log n)$-time algorithm that constructs a plane t-spanner for S, for $t \approx 10.02$, such that the degree of each point of S is bounded from above by 27, and the total edge length is proportional to the weight of a minimum spanning tree of S. These constants are all worst case constants that are artifacts of our proofs. In practice, we believe them to be much smaller. Previously, no algorithms were known for constructing plane t-spanners of bounded degree.

1 Introduction

Given a set S of n points in the plane and a real number $t > 1$, a *t-spanner* for S is a graph G with vertex set S such that any two vertices u and v are connected by a path in G whose length is at most $t \cdot |uv|$, where $|uv|$ is the Euclidean distance between u and v. If this graph has $O(n)$ edges, then it is a sparse approximation of the (dense) complete Euclidean graph on S.

Many algorithms are known that compute t-spanners with $O(n)$ edges that have additional properties such as bounded degree, small spanner diameter (i.e., any two points are connected by a t-spanner path consisting of only a small number of edges), low weight (i.e., the total length of all edges is proportional to the weight of a minimum spanning tree of S), and fault-tolerance; see, e.g., [1, 11, 12, 14], and the surveys [10, 16]. All these algorithms compute t-spanners for any given constant $t > 1$.

In this paper, we consider the construction of *plane* t-spanners. (Obviously, in order for a t-spanner to be plane, t must be at least $\sqrt{2}$.) It is known that the Delaunay triangulation is a t-spanner for $t = 2\pi/(3\cos(\pi/6))$, see [12]. Furthermore, in [5], it is shown that other plane graphs such as the minimum weight triangulation and the greedy triangulation are t-spanners, for some constant t. In [13], it is shown that, for any real number $r > 0$, a plane graph can be constructed from the Delaunay triangulation that is a t-spanner for $t = (1 + 1/r)2\pi/(3\cos(\pi/6))$ and whose total edge length is at most $2r + 1$ times the weight of a minimum spanning tree of S. Observe that all these spanners may have unbounded degree.

No algorithms are known that compute a plane t-spanner, for some constant t, of bounded degree. In this paper, we show how to compute such a spanner that, additionally, has low weight. That is, the main result of this paper is the following theorem.

*School of Computer Science, Carleton University, Ottawa, Ontario, Canada K1S 5B6. E-mail: {jit,michiel}@scs.carleton.ca. Research supported in part by NSERC.

†Institute of Information and Computing Sciences, Utrecht University, P.O.Box 80.089, 3508 TB Utrecht, the Netherlands. E-mail: joachim@cs.uu.nl. Supported by The Swedish Foundation for International Cooperation in Research and Higher Education.

Theorem 1.1 *There is an $O(n \log n)$-time algorithm that computes, when given a set S of n points in the plane and a constant $\varepsilon > 0$, a graph*

1. *that is plane,*
2. *that is a t-spanner for S, for $t = (\pi + 1)2\pi/(3\cos(\pi/6))(1 + \varepsilon)$,*
3. *in which each point of S has degree at most 27,*
4. *and whose weight is bounded from above by a constant times the weight of a minimum spanning tree of S.*

Our algorithm consists of the following steps. First, it computes the Delaunay triangulation of S, $DT(S)$, and a degree-3 spanning subgraph $BDS(S)$ of $DT(S)$ that includes the convex hull $CH(S)$ of S. This graph $BDS(S)$ partitions $CH(S)$ into (possibly degenerate) simple polygons, such that each point of S is on the boundary of at most three polygons. (This part of the algorithm is described in Section 2.)

Then, for each polygon P in the above partition, our algorithm processes the vertices of $DT(S) \cap P$ in breadth-first order, and prunes this part of the Delaunay triangulation such that each vertex of P has low degree. The resulting graph is a plane spanner for the vertices of P in the sense that any two vertices u and v of P are connected by a path whose length is at most a constant times the length of a shortest path between u and v that is completely contained in P. By combining all the spanners for each of the polygons, we get a plane spanner of bounded degree. (This part of the algorithm is described in Section 3.)

In the last step of the algorithm (see Section 4), we run the greedy algorithm (see [11]) on the graph obtained in Section 3. This results in a graph for which the claims in Theorem 1.1 hold.

2 Computing a degree-3 plane spanning graph (SpanningGraph)

The aim of this section is, given a set S of n points in the plane, to construct a low-degree plane spanning graph $BDS(S)$ of S that contains the convex hull of S. The algorithm presented in this section will be used as a pre-processing step in the main algorithm, as described in Section 1. The following lemma explains why we are interested in computing such a spanning graph.

Lemma 2.1 *Let $BDS(S)$ be a plane spanning graph of the set S that contains the convex hull of S, and let d be the maximum degree of any vertex of $BDS(S)$. The graph $BDS(S)$ partitions the convex hull of S into (possibly degenerate) simple polygons such that*

1. *the union of the vertices of all these polygons is equal to the set S,*
2. *no point of S is contained in the interior of any of these polygons, and*
3. *each point of S is on the boundary of at most d polygons.*

Degree-5 plane spanning subgraph. Let G be the union of the convex hull of S and a minimum spanning tree T of S. Then G is clearly a plane spanning graph. It is well known that each vertex of T has degree at most six, and the angle between any two adjacent edges of T is at least $60°$. Therefore, G has degree less than or equal to six. In [15], it is shown that there always exists a minimum spanning tree of degree at most five. By using this spanning tree T, the graph G has degree at most five.

Degree-3 plane spanning subgraph. Consider an arbitrary triangulation Δ of S. Barnette [2] has shown that every 3-connected planar graph contains a spanning tree of maximum degree at most three; his proof of this result implies an $O(n \log n)$-time algorithm for computing such a tree. If we apply this to Δ, then we obtain a plane degree-3 spanning tree of S. However, by adding the convex hull of S to this tree, we again obtain a degree-5 plane spanning graph of S that contains the convex hull of S.

We now show how to improve the degree to three. Our construction uses the *canonical ordering* of de Fraysseix *et al.* [8]. Let Δ be an arbitrary triangulation of S, and let $(v_1, v_2, \dots, v_n)$ be a numbering of the points of S. For any i, $1 \leq i \leq n$, let Δ_i denote the subgraph of Δ induced by the vertices $v_1, v_2, \dots, v_i$, and let C_i be the outer face of Δ_i. We say that this numbering is a canonical ordering of Δ, if the following conditions are satisfied:

- v_1 and v_2 are adjacent in Δ and are both on the convex hull of S.
- For all i, $3 \leq i \leq n$, the following holds.
  - the subgraph Δ_i is 2-connected.
  - The point v_i is in the interior of the outer face C_{i-1} of Δ_{i-1}.
  - Let N_i be the set of all vertices of Δ_{i-1} that are connected by an edge, in Δ, to v_i. The set N_i contains at least two elements, and forms a path on the boundary of C_{i-1}.

A canonical ordering always exists. It can be computed, in reverse order, as follows. Start with the complete triangulation Δ, and successively remove a vertex from the boundary of the outer face that is incident to exactly two vertices on the outer face. It is easy to show that such a vertex always exists. The time to compute this canonical ordering is $O(n)$; see [8].

We apply this to an arbitrary triangulation Δ of S. Consider a canonical ordering $(v_1, v_2, \dots, v_n)$ of Δ. For each i, $3 \leq i \leq n$, we will compute a subgraph G_i of Δ. During this construction, we maintain the following invariant.

Invariant: The graph G_i is a spanning graph for $\{v_1, v_2, \dots, v_i\}$, its degree is at most three, and it contains all edges of Δ that form the boundary of C_i.

We define the graph G_3 to be the triangle spanned by v_1, v_2, and v_3. Let i be an integer with $4 \leq i \leq n$, and assume we have constructed the graph G_{i-1}. Consider the set N_i of all vertices of Δ_{i-1} that are connected by an edge, in Δ, to v_i. Recall that $|N_i| \geq 2$ and N_i forms a path on the boundary of C_{i-1}. Let $u_1, \dots, u_k$ be all vertices of N_i, in the order in which they occur on this path. We distinguish two cases.

$k = 2$**:** We obtain G_i from G_{i-1} by removing the edge (u_1, u_2) and adding the edges (v_i, u_1) and (v_i, u_2). (See the top part of Figure 1.)

$k > 2$**:** We obtain G_i from G_{i-1} by removing the edges (u_1, u_2) and (u_{k-1}, u_k) and adding the edges (v_i, u_1), (v_i, u_2) and (v_i, u_k). (See the bottom part of Figure 1.)

It is clear that the invariant is maintained during this construction. Hence, the graph $BDS(S) := G_n$ is a plane spanning graph of S of maximum degree at most three. Observe that C_n is equal to the outer face of Δ. Therefore, the boundary of C_n is the convex hull of S. Hence, the invariant implies that $BDS(S)$ contains the convex hull of S.

We denote the entire algorithm for computing $BDS(S)$, when given the triangulation Δ, by SpanningGraph(Δ). An example is given in Figure 2. The following lemma summarizes the result of this section.

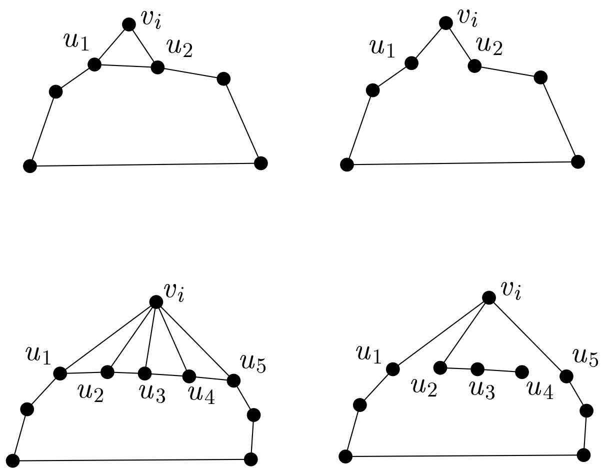

Figure 1: *Illustrating the construction of a degree-3 plane spanning graph.*

Lemma 2.2 *Given a triangulation Δ of a set S of n points in the plane, we can compute, in $O(n)$ time, a plane spanning graph $BDS(S)$ that contains the convex hull of S and in which each vertex has degree at most three.*

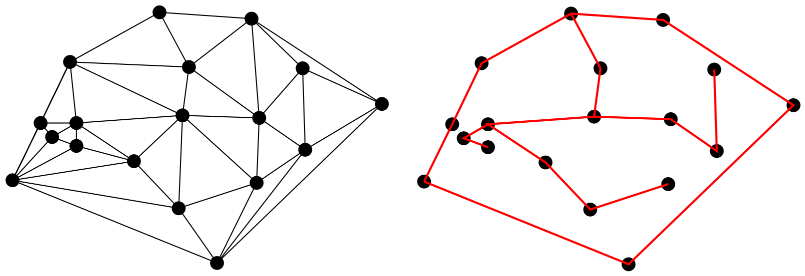

Figure 2: *A triangulation Δ (on the left), and the output of algorithm* SpanningGraph(Δ) *(on the right).*

2.1 Algorithm TransformPolygon

The polygons obtained from algorithm SpanningGraph(Δ) may be degenerate, i.e., there may be vertices of degree one or cut vertices in any such polygon. An easy way to handle this problem is to symbolically transform each degenerate polygon P' into a non-degenerate polygon P. This is done by doubling edges adjacent to degenerate vertices of P' and then running an Euler tour. Figure 3 illustrates how this transformation is performed on a degenerate vertex of degree 3. All other degenerate vertices can be handled in a similar fashion.

This transformation does not affect the complexity of the spanner algorithm if the vertices are only split symbolically. We denote the algorithm that transforms a degenerate polygon P' into a non-degenerate polygon P by TransformPolygon(P').

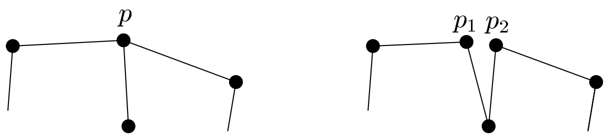

Figure 3: *Illustrating algorithm* TransformPolygon

3 Computing a spanner of a simple polygon (PolygonSpanner)

The idea behind our algorithm is to start with the Delaunay triangulation $DT(S)$ of a point set S which is known to be a t-spanner with $t \leq 2\pi/(3\cos(\pi/6))$ (shown in [12]). Remove edges from $DT(S)$ until we are left with a graph $PS(S)$ having two properties:

1. every vertex in $PS(S)$ has bounded degree,

2. for every edge $xy \in DT(S)$, there is a path between x and y in $PS(S)$ whose length is at most $\pi + 1$ times $|xy|$.

These two properties combined imply that $PS(S)$ is a plane graph, every vertex has bounded degree and it is a $((\pi+1)2\pi/(3\cos(\pi/6)))$-spanner. To achieve the first property, we construct the maximum degree-3 spanning subgraph, $BDS(S)$, of the Delaunay triangulation as described in the previous section. Of course, this subgraph is not necessarily a constant spanner, so too many edges have been removed at this point. Therefore, we need to add some carefully chosen edges in order to maintain the first property and achieve the second. Our starting point is $BDS(S)$. As noted previously, $BDS(S)$ decomposes the plane into a set of simple polygons, $\{P_1, \ldots, P_k\}$. Now, each edge $xy \in DT(S)$ that is *not* in $BDS(S)$ must be a diagonal in some P_i. Thus, it is sufficient to show how to properly approximate all such edges within this set of polygons.

Let P be a simple polygon with vertex set V, and let $t > 1$ be a real number. A graph with vertex set V is called a *t-spanner for V w.r.t. P*, if for every pair of points $u, v \in V$ that form a diagonal of P, there exists a path between u and v in the graph, whose length is at most t times the length of $|uv|$.

In this section, we show how to compute, for some constant $t > 1$, a plane t-spanner for V w.r.t. P, whose degree is bounded by a constant. By constructing such a spanner for each polygon P_i, we get the desired spanner.

Let $G = (V, E)$ be the plane graph, where V is the set of vertices of P and E is the union of the edge set of P and the edges of the Delaunay triangulation of V contained in P. Let v_1 be an arbitrary vertex of V, and let $v_1, v_2, \ldots, v_n$ be the vertices of V according to a clockwise breadth-first ordering (bfs-order) of G with start vertex v_1. A clockwise breadth-first search of a graph is simply a standard breadth-first search with the additional constraint that when the children of a vertex are processed, they are processed in clockwise order. For any two vertices u and v of V, we write $u <_{bf} v$, if u strictly precedes v in the bfs-ordering. We extend this ordering to $\leq_{bf}$ in the obvious way.

We start with some properties of the bfs-numbering.

Let $\mathcal{T}(G, v_1)$ be the breadth-first search tree of G rooted at vertex v_1. The level of any vertex v in $\mathcal{T}(G, v_1)$ is denoted by $\ell(v)$. (The root is at level zero, i.e., $\ell(v_1) = 0$.)

Lemma 3.1 *Let v be any vertex in $\mathcal{T}(G, v_1)$ such that $v \neq v_1$.*

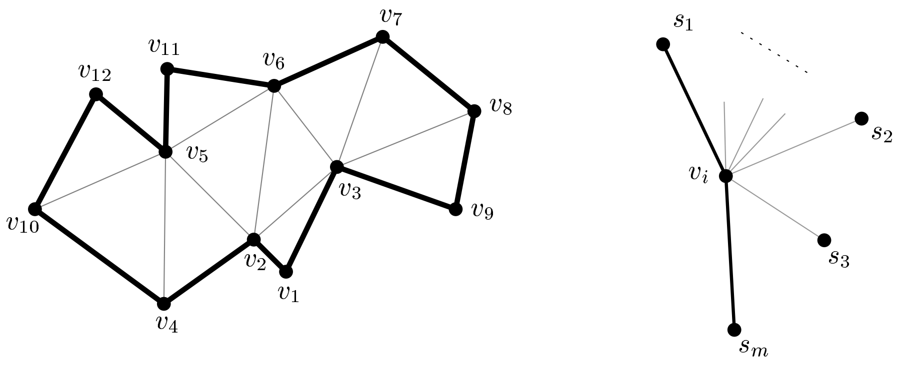

Figure 4: *Illustration of clockwise breadth-first search.*

---

a) Any Delaunay edge (v, w) that is not an edge of P partitions P into two simple polygons P_1 and P_2. We may assume w.l.o.g. that v_1 is a vertex of P_1. For every vertex u in P_2 with $u \neq v$ and $u \neq w$, we have $\min(\ell(v), \ell(w)) <_{bf} \ell(u)$ and $\max(\ell(v), \ell(w)) \leq_{bf} \ell(u)$.

b) The line segment between v and its parent u in $\mathcal{T}(G, v_1)$ partitions P into two simple polygons P_1 and P_2 such that all children of u that precede v are in P_1 and all siblings that succede v are in P_2. The vertex v can be adjacent to at most one child of u that precedes v.

Proof: (a) The claim holds since either v or w must be an ancestor to all vertices in P_1. (b) Follows since the siblings of the parent of v are ordered in clockwise order, see Fig. 4. $\square$

The sequence of vertices adjacent to a vertex v, sorted in clockwise order around v, will be denoted by $\mathcal{N}(v)$ and will be called the *ordered neighborhood* of v.

Lemma 3.2 *The vertices in $\mathcal{N}(v)$ preceding v in the bfs-ordering are the parent of v, denoted $p(v)$, and at most two vertices in $\mathcal{N}(v) \cap \mathcal{N}(p(v))$.*

Proof: Let v_1 and v_2 be the two vertices adjacent to v and $p(v)$. The quadrangle $(p(v), v_1, v, v_2)$ partitions P into five simple polygons, denoted $P_1, \ldots, P_5$ as illustrated in fig. 5. P_2 and P_3 are bounded by $(p(v), v_1)$ and $(p(v), v_2)$ respectively and hence cannot contain any vertices connected to v. Now consider P_4 and P_5 with boundary edge (v_1, v) and (v_2, v) respectively. The proof for both P_4 and P_5 are identical. We consider P_4. There are two cases:

- If $\ell(v) = \ell(v_1)$ then every vertex u in P_4, except v and v_1, must be a successor of v since $\ell(u) > \ell(v)$ according to Lemma 3.1a.
- If $\ell(v) > \ell(v_1)$ then every point u in P_4 must be a successor of v since $p(v) <_{bf} v_1$, otherwise $p(v)$ could not be the parent of v.

The lemma follows. $\square$

Now we are ready to present the algorithm that computes a bounded-degree plane spanner for V w.r.t. P. Recall that $G = (V, E)$ is the plane graph, where E is the union of the edge set of P

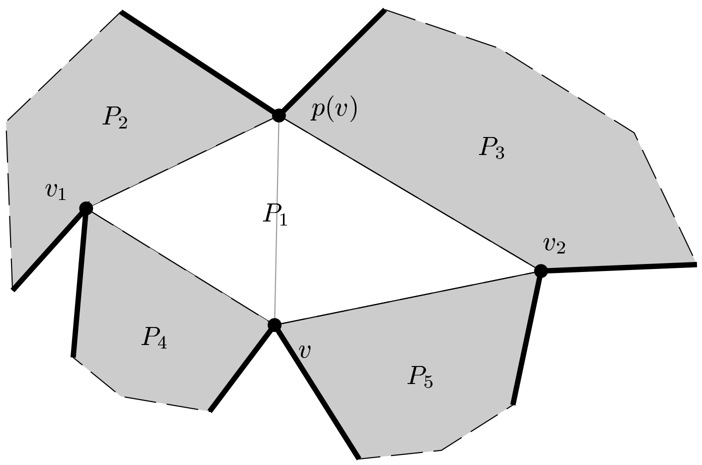

Figure 5: *Illustration for Lemma 3.2*

and the edge set consisting of all edges of the Delaunay triangulation of V that are contained in P. Our algorithm processes all the vertices of G according to their bfs-ordering starting with v_1.

Initialize E' to be the set of edges of P. At the end of the processing, E' will be the edges of the desired spanner. Let $s_1, s_2, \ldots, s_m$ be the ordered neighborhood $\mathcal{N}(v_1)$ of v. That is, $s_1, s_2, \ldots, s_m$ are all vertices that are connected by an edge of E to v_1. These vertices occur in clockwise order around v_1. We assume that the edges (s_1, v_1) and (s_m, v_1) are edges of the polygon P. The following two steps are performed in the processing of v_1.

First, the interior angle at v_1 is divided into a minimum number of intervals of at most $90°$. For each interval, add the shortest edge in E that is connected to v_1 to the edge set E'. Second, add to E' all edges (s_j, s_{j+1}), $1 \leq j < m$. This terminates the processing of vertex v_1.

A generic processing step is similar. Assume that vertex v_i of V is next to be processed. Let $s_1, \ldots, s_m$ be the ordered neighborhood of v_i. We assume that (s_1, v_i) and (s_m, v_i) are edges of P. It follows from Lemma 3.2 that the parent of v_i has been processed and possibly at most two vertices s_a, s_b in $\mathcal{N}(v_i) \cap \mathcal{N}(p(v_i))$. The remainder of the ordered neighborhood of v_i needs to be processed. Let $\alpha_1(v_i) = \angle s_1 v_i s_a$ and $\alpha_2(v_i) = \angle s_b v_i s_m$. The following two steps are performed.

First, we divide the interior angle $\angle s_1 v_i s_a$ into a minimum number of intervals of at most $90°$. The same is done for the angle $\angle s_b v_i s_m$. For each interval, we add to the set E' the shortest edge in E that connects v_i to a vertex that has not been processed. Secondly, we add all edges (s_j, s_{j+1}), $1 \leq j < m$, to E'. Vertex v_i has now been processed.

We continue processing vertices in this way until all vertices have been processed. The output of this algorithm is the graph $G' = (V, E')$. We will refer to this algorithm as PolygonSpanner(P).

3.1 Analysis of algorithm PolygonSpanner

In this section, we will show that G' is a spanner with respect to P and that the degree of any vertex of G' is bounded by a constant.

Lemma 3.3 *Let v be any vertex of P. At the moment when v is being processed by algorithm* PolygonSpanner, *there are at most five edges in E' that are incident to v.*

Proof: The proof uses Lemma 3.2. First observe that the parent $p(v)$ of v is the first vertex incident to v that is being processed. The edges incident to v that the algorithm may add to E'

while processing $p(v)$ are the Delaunay edge $(v, p(v))$ and the edges on the ordered neighborhood of $p(v)$; hence a total of three edges. Assume that v is connected to the vertices v_1 and v_2 on the ordered neighborhood of $p(v)$. These are the only two vertices adjacent to v that can be processed before it. However, since $p(v)$ is in $\mathcal{N}(v) \cap \mathcal{N}(v_1)$ and also in $\mathcal{N}(v) \cap \mathcal{N}(v_2)$, at most two edges adjacent to v can be added to E' while processing v_1 and v_2. Hence, when v is being processed, there are at most five edges in E' incident to v. □

Once a vertex has been processed, no additional edges adjacent to it are ever added. Together with the above results, it follows that the degree, in the graph G', of any vertex $v \in V$ is bounded from above by

$$5 + \lceil \alpha_1(v)/90^\circ \rceil + \lceil \alpha_2(v)/90^\circ \rceil + 2 \leq 9 + \alpha_1(v)/90^\circ + \alpha_2(v)/90^\circ < 13,$$

since $\alpha_1(v) + \alpha_2 v < 360^\circ$. Hence, the degree is at most 12. This immediately gives a bound of at most 36 if all polygons are processed this way since each vertex can participate in at most 3 polygons. We show how to reduce this upper bound in Section 4.

Lemma 3.4 *For any edge $(u, v) \in DT(V)$ inside P, there exists a path in G' between u and v whose length is at most $\pi + 1$ times the edge $|uv|$.*

Proof: Consider an arbitrary edge $e = (u, v)$ in the Delaunay triangulation of S that is contained in P. If $e \in G'$, the lemma holds, so assume that $e \notin G'$.

Assume w.l.o.g. that $u <_{bf} v$. It follows from the algorithm that there must exist a vertex v' in the ordered neighborhood of u such that $\angle v'uv < 90^\circ$ and $(v', u) \in E'$. Let $v' = s_1, s_2, \ldots, s_k = v$ be this sequence of vertices in the ordered neighborhood of u from v' to v. Since $(v', u) \in E'$ is shorter than (u, v) by definition, we only need to show that the segment (v, v') is well approximated in G' in terms of (u, v).

Consider the polygon P', which is a sub-polygon of P, consisting of the vertices $u, s_1, \ldots, s_k$. The shortest path in P' from v' to v, denoted $S_{P'}(v', v)$, consists of diagonals of P' and is contained in the triangle u, v, v'. By convexity, the length of $S_{P'}(v', v)$ is at most 2 times $|uv|$.

An edge of $S_{P'}(v', v)$ has as endpoints two vertices in the neighborhood of u. Let (s_i, s_j) be an arbitrary edge in $S_{P'}(v', v)$. Let $D_{P'}(s_i, s_j)$ be the sequence of edges between s_i and s_j in the ordered neighborhood of u. This path is in G'. We bound the length of $D_{P'}(s_i, s_j)$ in terms of $|s_i s_j|$.

In [3], it is shown (by modifying an argument in [9]), that the length of $D_{P'}(s_i, s_j)$ is at most $\pi/2$ times $|s_i s_j|$, provided that (i) the straight-line segment between s_i and s_j lies outside the Voronoi region induced by u, and (ii) that the path lies on one side of the line through s_i and s_j.

Condition (ii) holds trivially. We now show that condition (i) also holds. The vertices u, s_i and s_j form a triangle where $\angle s_i u s_j < 90^\circ$. This implies that for any point x on the segment $[s_i, s_j]$, $\min\{|xs_i|, |xs_j|\} < |xu|$. This means that u cannot be in the Voronoi region induced by u. Therefore, $D_{P'}(s_i, s_j)/|s_i s_j| \leq \pi/2$.

Putting it all together, it follows that the shortest path in G' between v' and v has length at most $\pi|uv|$. This implies that the shortest path in G' between u and v is at most $(\pi + 1)|uv|$, hence the lemma follows. □

4 The complete algorithm and its analysis

In this section, we present the algorithm which will prove Theorem 1.1. The algorithm is given in Figure 6. Step 7 refers to the greedy algorithm of Gudmundsson *et al.* [11]. This algorithm takes as input an arbitrary spanner $PS(S) = (S, E)$ and a real constant $\varepsilon > 0$, and computes a $(1+\varepsilon)$-spanner $G = (S, E')$ of $PS(S)$, such that $E' \subseteq E$ and the weight of E' is bounded by a constant times the weight of a minimum spanning tree of S. In the rest of this section, we will prove that for the output of algorithm PlanarSpanner(S, ε), the claims in Theorem 1.1 hold.

Algorithm PlanarSpanner(S, ε)
1. $DT \leftarrow$ Delaunay triangulation of S
2. $BDS(S) \leftarrow$ SpanningGraph(DT) (see Section 2)
3. **for each** polygon P' in the partition induced by $BDS(S)$ **do**
4. $P \leftarrow$ TransformPolygon(P') (see Section 2.1)
5. $G_P \leftarrow$ PolygonSpanner(P) (see Section 3)
6. $PS(S) \leftarrow$ union of all G_P
7. $G \leftarrow$ GreedySpanner$(PS(S), \varepsilon)$ (see [11])
8. output G

Figure 6: *The construction of a plane sparse spanner.*

Let us first consider the running time of algorithm PlanarSpanner. Step 1 takes $O(n \log n)$ time (see [7]) and, by Lemma 2.2, Step 2 takes $O(n)$ time. By the results of Section 3, Steps 4 and 5 take $O(|P'|)$ time for each polygon P' in the partition induced by the spanning graph $BDS(S)$. Since each point of S is a vertex of at most three such polygons P' (see Lemma 2.1), the total time for the for-loop is $O(n)$. Since each graph G_P contains $O(|P|)$ edges, Step 6 takes $O(n)$ time. Finally, by the results in [11], Step 7 takes $O(n \log n)$ time. Hence, the total time of the algorithm is $O(n \log n)$.

Since the output G is a subgraph of $PS(S)$, and $PS(S)$ is a subgraph of the Delaunay triangulation DT of S, it is clear that G is plane, proving the first claim in Theorem 1.1.

To prove the second claim in Theorem 1.1, first observe that DT is a $(2\pi/(3\cos(\pi/6))$-spanner for S, see [12]. Consider any two distinct points u and v of S, and let Q be a path in DT between u and v having length at most $(2\pi/(3\cos(\pi/6)) \cdot |uv|$. Consider an arbitrary edge (x, y) on Q that is not an edge in $PS(S)$. This edge must be a diagonal in some polygon P'. By Lemma 3.4, there is a path in $PS(S)$ between x and y whose length is at most $(\pi + 1)|xy|$. Therefore, there is a path in $PS(S)$ between u and v of length at most $(\pi + 1)(2\pi/(3\cos(\pi/6))|uv|$. Hence, $PS(S)$ is a $((\pi + 1)(2\pi/(3\cos(\pi/6)))$-spanner for S. The results in [11] imply that G is a t-spanner for S, for $t = (\pi + 1)(2\pi/(3\cos(\pi/6))(1 + \varepsilon)$.

Next, we prove the third claim in Theorem 1.1. As described in Section 2.1, a degenerate polygon P' is transformed in Step 4 to a simple polygon P by doubling some edges. It is easily seen that in this process, any original point u is replaced by at most three new vertices u_1, u_2, and u_3. An analysis similar to the one just above Lemma 3.4 shows that when these three vertices are merged again, the degree of u in $PS(S)$ is at most $3 \cdot 5 + 3 + \lceil \frac{\alpha_1(u_1)}{90^\circ} \rceil + \lceil \frac{\alpha_2(u_1)}{90^\circ} \rceil + \lceil \frac{\alpha_1(u_2)}{90^\circ} \rceil + \lceil \frac{\alpha_2(u_2)}{90^\circ} \rceil + \lceil \frac{\alpha_1(u_3)}{90^\circ} \rceil + \lceil \frac{\alpha_2(u_3)}{90^\circ} \rceil$. Since all these six interior angles sum up to 360°, the degree of u is at most 27.

The fourth claim in Theorem 1.1 follows from results in [4, 6]: Since $PS(S)$ is a $((\pi+1)(2\pi/(3\cos(\pi/6)))$-

spanner for S, the graph G produced by the greedy algorithm has a weight that is proportional to the weight of a minimum spanning tree of S.

5 Concluding remarks

The results in this paper generalize to the *constrained* case, i.e., when the set S is a set of line segments rather than a point set, and edges of the spanner are not allowed to intersect any edge of S. The details will be presented in the full paper.

There are additional properties that would be interesting to consider, for example minimizing the diameter of the spanner. Below we show that this property cannot be obtained for plane spanners.

Lemma 5.1 *For any real number $t > 1$, there exists a set S of n points and two points $u, v \in S$ such that for any plane t-spanner it holds that any t-spanner path between u and v contains at least $n/2 + 1$ edges.*

Proof: The construction of the point set is illustrated in Figure 7: For simplicity we assume that n is even. Place u and v on a horizontal line such that the horizontal distance between u and v is $t \cdot \frac{n-2}{2}$.

□

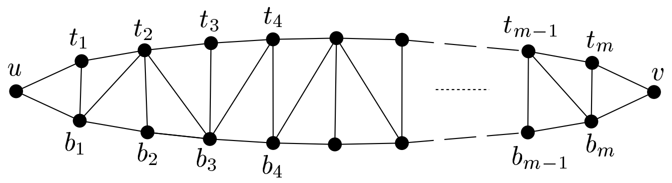

Figure 7: *Illustrating the proof of Lemma 5.1*

Acknowledgements

The authors wish to thank Anil Maheshwari, Pat Morin and David Wood for fruitful discussions on this topic.

References

[1] S. Arya, G. Das, D. M. Mount, J. S. Salowe, and M. Smid. Euclidean spanners: short, thin, and lanky. In *Proc. 27th Annu. ACM Sympos. Theory Comput.*, pages 489–498, 1995.

[2] D. Barnette. Trees in polyhedral graphs. *Canadian Journal of Mathematics*, 18:731–736, 1966.

[3] P. Bose and P. Morin. Online routing in triangulations. In *Proc. 10th Annu. Internat. Sympos. Algorithms Comput.*, volume 1741 of *Lecture Notes Comput. Sci.*, pages 113–122. Springer-Verlag, 1999.

[4] G. Das, P. Heffernan, and G. Narasimhan. Optimally sparse spanners in 3-dimensional Euclidean space. In *Proc. 9th Annu. ACM Sympos. Comput. Geom.*, pages 53–62, 1993.

[5] G. Das and D. Joseph. Which triangulations approximate the complete graph? In *Proc. International Symposium on Optimal Algorithms*, volume 401 of *Lecture Notes Comput. Sci.*, pages 168–192. Springer-Verlag, 1989.

[6] G. Das, G. Narasimhan, and J. Salowe. A new way to weigh malnourished Euclidean graphs. In *Proc. 6th ACM-SIAM Sympos. Discrete Algorithms*, pages 215–222, 1995.

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

[8] H. de Fraysseix, J. Pach, and R. Pollack. How to draw a planar graph on a grid. *Combinatorica*, 10:41–51, 1990.

[9] D. P. Dobkin, S. J. Friedman, and K. J. Supowit. Delaunay graphs are almost as good as complete graphs. *Discrete Comput. Geom.*, 5:399–407, 1990.

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

[11] J. Gudmundsson, C. Levcopoulos, and G. Narasimhan. Improved greedy algorithms for constructing sparse geometric sp anners. In *Proc. 7th Scand. Workshop Algorithm Theory*, volume 1851 of *Lecture Notes Comput. Sci.*, pages 314–327, Berlin, 2000. Springer-Verlag.

[12] J. M. Keil and C. A. Gutwin. Classes of graphs which approximate the complete Euclidean gr aph. *Discrete Comput. Geom.*, 7:13–28, 1992.

[13] C. Levcopoulos and A. Lingas. There are planar graphs almost as good as the complete graphs and almost as cheap as minimum spanning trees. *Algorithmica*, 8:251–256, 1992.

[14] C. Levcopoulos, G. Narasimhan, and M. Smid. Improved algorithms for constructing fault-tolerant spanners. *Algorithmica*, 32:144–156, 2002.

[15] C. Monma and Subhash Suri. Transitions in geometric minimum spanning trees. *Discrete Comput. Geom.*, 8:265–293, 1992.

[16] 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 Publishers, Amsterdam, 2000.