

# Steiner tree in planar graphs: An $O(n \log n)$ approximation scheme with singly-exponential dependence on epsilon

Glencora Borradaile\*, Philip N. Klein\*\*, and Claire Mathieu

Brown University, Providence RI 02912, USA.  
{glencora, philip, claire}@cs.brown.edu

**Abstract.** We give an algorithm that, for any  $\epsilon > 0$ , any undirected planar graph  $G$ , and any set  $S$  of nodes of  $G$ , computes a  $(1 + \epsilon)$ -optimal Steiner tree in  $G$  that spans the nodes of  $S$ . The algorithm takes time  $O(2^{\text{poly}(1/\epsilon)} n \log n)$ .

## 1 Introduction

The Steiner problem in graphs is a fundamental and well-studied optimization problem: given a graph with edge lengths and a set of terminals, find a minimum-length connected subgraph that includes all the terminals. The problem is NP-hard [11] (even in planar graphs [8]) and is max SNP-complete in general graphs [5]. Much work [22, 14, 24, 23, 15, 26, 2, 25, 17, 12, 10, 21] has gone into obtaining constant-factor approximation algorithms. There has also been work [1, 16, 20] on approximation schemes for the case of Euclidean plane (or more generally low-dimensional Euclidean space). In [6], we gave an  $O(n \log n)$  approximation scheme for Steiner tree in planar graphs; more precisely, we showed that for any  $\epsilon > 0$ , there is an  $O(n \log n)$  algorithm that returns a solution whose length is at most  $1 + \epsilon$  times optimal. However, the constant factor for this algorithm is triply exponential in  $1/\epsilon$ . In this paper, we give another  $O(n \log n)$  approximation scheme, one for which the constant factor is singly exponential in  $1/\epsilon$ :

**Theorem 1.** *For any  $\epsilon > 0$ , there is an algorithm that, given a planar graph  $G$  with edge lengths and a set  $S$  of vertices of  $G$ , finds a Steiner tree that spans  $S$  and whose length is at most  $1 + \epsilon$  times the length of the optimal Steiner tree spanning  $S$ . The running time is  $O(2^{\text{poly}(1/\epsilon)} n + n \log n)$  where  $n$  is the number of vertices of  $G$ .*

The previous approximation scheme fit into a framework given in [13]. What the framework required to yield an approximation scheme (and what was provided in [6]) was a kind of “spanner” result: an algorithm that, for a planar graph  $G_{in}$

---

\* Supported by a Kanellakis Fellowship and a Brown University Dissertation Fellowship.

\*\* Partially supported by NSF Grant CCF-0635089.

and set of terminals  $Q$ , would return a “short” subgraph of  $G_{in}$  that approximately preserved the value of the optimum. The approximation scheme then follows rather directly from the framework. However, the spanner result had a doubly-exponential dependence on  $1/\epsilon$ , which led to the triply-exponential dependence of the final approximation scheme’s running time.

We overcome this deficiency using two ideas. One of the exponentials came from a theorem in which we showed how to reduce the complexity of a subtree of the optimal Steiner tree without increasing its cost too much. In this paper, we prove the same theorem but with a polynomial instead of an exponential (see Theorem 4). This idea by itself, if plugged back into [6], would yield doubly exponential dependence on  $1/\epsilon$ .

The other idea is more global, and perhaps will turn out to be more generally applicable for obtaining approximation schemes in planar graphs. The first step of the spanner construction consisted of finding a short grid-like subgraph  $MG$  of the input graph  $G_{in}$  that contains every terminal. In this paper, we use the term *brick* to refer the subgraph consisting of a face of  $MG$  and the subgraph of  $G_{in}$  embedded inside it.

In the new approximation scheme, we also start by finding  $MG$ . We then decompose  $MG$  into “parcels” with short boundaries such that each parcel has low carving-width (a relative of tree-width). Of course,  $MG$  is not the original graph; it is missing the bricks. We add back the bricks but connect them to  $MG$  via a small number of “portal edges”. We add some new terminals and, for each parcel-plus-bricks, find an optimal Steiner tree using dynamic programming (exploiting the low carving-width of the parcel and the small number of portal edges). We prove that the union of all these trees is not much longer than the optimal Steiner tree in the original graph. The base case for the dynamic program uses an exact algorithm by Erickson et al. [7] for the special case where all terminals are on the boundary of an embedded planar graph. We summarize their result in the following theorem:

**Theorem 2.** [7] *Let  $G$  be a planar embedded graph and  $Q$  be a set of  $k$  terminals that all lie on the boundary of a single face. Then there is an algorithm<sup>1</sup> to find an optimal Steiner tree of  $Q$  in  $G$  in time  $O(nk^3 + (n \log n)k^2)$ .*

We can replace the term  $n \log n$  by  $n$  using the linear-time planarity-exploiting shortest-path algorithm of [9], obtaining a running time of  $O(nk^3)$ .

### 1.1 Notation

We assume without loss of generality that the input graph  $G_{in}$  is planar embedded and has degree at most three. The input graph has positive edge-lengths  $\ell(\cdot)$ . For a set  $A$  of edges, we use  $\ell(A)$  to denote  $\sum_{e \in A} \ell(e)$ . We take  $Q$  as the set of terminal vertices.

<sup>1</sup> This algorithm has been generalized by Bern [3] and by Bern and Bienstock [4] to handle some additional special cases, e.g. where the terminals lie on a constant number of faces. Provan [19, 18] used the same approach to give exact and approximate algorithms for some geometric special cases.

For notational convenience, we prove a slightly weaker version of main theorem (Theorem 1). We show that, for any  $\epsilon > 0$ , there is an  $O(n \log n)$  algorithm to find a Steiner tree whose length is at most  $(1 + c\epsilon)\text{OPT}(G_{in}, Q)$ , where  $c$  is a constant and  $\text{OPT}(G, Q)$  denotes the length of the Steiner minimal tree that spans  $Q$  in graph  $G$ . To prove Theorem 1, for any given  $\bar{\epsilon} > 0$ , we set  $\epsilon = \bar{\epsilon}/c$  and invoke the algorithm.

The *boundary of a face* of a planar embedded graph is the set of edges adjacent to the face; it does not always form a simple cycle (Figure 2(a)). The *boundary*  $\partial H$  of a planar embedded graph  $H$  is the set of edges bounding the infinite face. An edge is *strictly enclosed* by the boundary of  $H$  if the edge belongs to  $H$  but not to  $\partial H$ .

Graphs are identified with sets of edges, thus a subgraph  $H$  of a graph  $G$  is also considered a subset of the edges of  $G$ . The set of vertices that are endpoints of edges in  $H$  is denoted  $V(H)$ . For a tree  $T$  and vertices  $x, y \in V(T)$ , we denote the unique simple  $x$ -to- $y$  path in  $T$  by  $T[x, y]$ . In particular, if  $T$  is a path then  $T[x, y]$  is the  $x$ -to- $y$  subpath. We denote the length of the shortest  $x$ -to- $y$  path in  $G$  as  $\text{dist}_G(x, y)$ .

For a connected planar embedded graph  $G$ , there is another connected planar embedded graph denoted  $G^*$ . The faces of  $G$  are the vertices of  $G^*$ ; the edges of the two graphs are identified. We refer to  $G$  as the *primal* graph and to  $G^*$  as the *dual*.

## 2 Algorithm

### 2.1 Mortar Graph

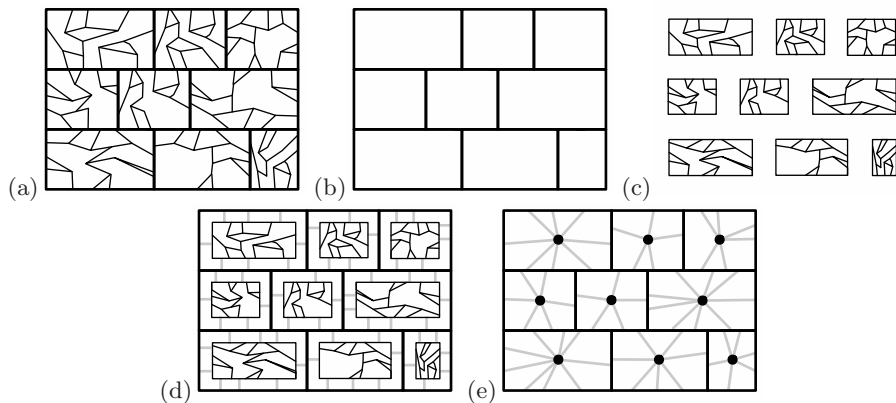
We first find a connected grid-like subgraph of the input graph  $G_{in}$ , based on the set  $Q$  of terminals and the given precision  $\epsilon$ . The  $O(n \log n)$ -construction is given in [6]. We call the subgraph the *mortar graph* and denote it  $MG$  (see Figure 1(b)). The mortar graph spans every terminal in  $Q$  and has length at most  $5\epsilon^{-1} \cdot \text{OPT}(G_{in}, Q)$ . We defer the remaining properties of  $MG$  to Section 3, where they are needed for the proof of correctness.

*Step 1:* Construct the mortar graph,  $MG$ .

### 2.2 Bricks

Each face  $f$  of the mortar graph that strictly encloses at least one edge of  $G_{in}$  defines a graph called a *brick*. The brick consists of the edges of  $G_{in}$  that are enclosed by the boundary  $\partial f$  of  $f$ . This boundary is a cycle of edges, possibly with repetition if some edges occur twice in the boundary (an example of such a situation is shown on Figure 2). We duplicate the repeated edges as follows:

Cut the original graph  $G_{in}$  along  $\partial f$ , duplicating the edges you cut along (and replicating the vertices), and define the brick to be the subgraph of  $G_{in}$  embedded inside that cycle, including the boundary edges according to their



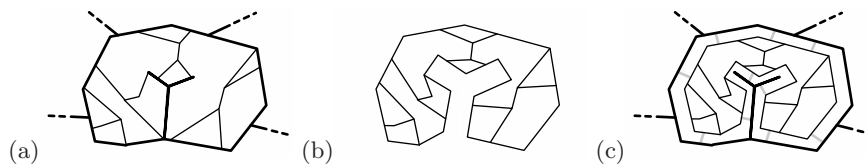
**Fig. 1.** (a) A fragment of an input graph  $G_{in}$ . The bold edges belong to the mortar graph  $MG$ , the corresponding fragment of which is shown in (b). The corresponding bricks are shown in (c), and the corresponding fragment of the portal-connected graph,  $\mathcal{B}^+(MG)$ , appears in (d). The portal edges are grey. (e)  $\mathcal{B}^+(MG)$  with the bricks contracted to *brick vertices*.

multiplicity in  $\partial f$ . That is, if an edge occurs twice in the boundary of the face, then there are two copies of that edge in the corresponding brick.

*Step 2:* Compute the set of bricks,  $\mathcal{B}$ .

It is easy to see that Step 2 takes  $O(n)$ .

The boundary  $\partial B$  of a brick  $B$  is the simple cycle of boundary edges. The corresponding face of  $MG$  is called the *mortar boundary* of  $B$ . Each edge of the mortar graph occurs at most twice in the disjoint union of the boundaries of the bricks. Since we defined bricks corresponding only to non-empty faces, every brick contains at least one edge not belonging to  $MG$ . Figure 1(c) is an example of the set of bricks corresponding to the mortar graph of Figure 1(b). The construction of a brick is illustrated in Figures 2(a) and (b).



**Fig. 2.** Construction of a brick: (a) The boundary of a face  $f$  of  $MG$  is a cycle of edges (thick edges), possibly with repetition (i.e. an edge can occur twice in the boundary). The light edges are those in the interior of  $f$  in  $G_{in}$ . (b) We obtain the corresponding brick via Step 2. The resulting brick  $B$  has boundary  $\partial B$ . (c) A brick, copied.

### 2.3 Portals

For portal selection, we use a parameter  $\theta(\epsilon) = 2\alpha(\epsilon)5\epsilon^{-2}$  that depends on a value  $\alpha(\epsilon)$  that in turn comes out of Theorem 3. Portals are selected greedily as in [6] to satisfy:

*Property 1.* For any vertex  $x$  on  $\partial B$ , there is a portal  $y$  such that the  $x$ -to- $y$  subpath of  $\partial B$  has length at most  $\ell(\partial B)/\theta$ .

*Step 3:* For each brick  $B$ , designate  $\theta$  vertices of  $\partial B$  as *portals*.

We additionally require that one portal be the endpoint of an edge that is strictly enclosed in the brick (this, in addition to the assumption that our input graph is degree three, allows us to build a binary recursion tree for the dynamic program).

### 2.4 Portal-connected graph and the operation $\mathcal{B}^+$

In preparation for stating our Structure Theorem, we define an operation called *brick insertion*. For any subgraph  $MG$  of  $G$ , we derive a planar embedded graph  $\mathcal{B}^+(G)$  as follows. For each face  $f$  of  $G$  corresponding to a brick  $B$ , embed a copy of  $B$  inside the face  $f$ , and, for each portal vertex  $v$  of  $B$ , connect  $v$  in the brick to the corresponding vertex in  $f$ , using a zero-length artificial edge (Figure 2(c)). We refer to the artificial edges as *portal edges*. This step is illustrated in Figure 1(d). We refer to  $\mathcal{B}^+(MG)$  as the *portal-connected graph*, and we denote it by  $\mathcal{B}^+(MG)$ . Intuitively, this graph is almost the same as the input graph  $G_{in}$ , except that artificial cost-zero separations have been added so that paths that connect vertices strictly enclosed by faces of the mortar graph to outside vertices are forced to go through the portals.

If a vertex of  $MG$  is a terminal of  $Q$ , we do not consider its copy on the brick to be a terminal vertex. Thus a brick has no terminals; this helps in the design of the dynamic program (Section 2.7).

The following lemma follows directly from the fact that each portal edge in  $\mathcal{B}^+(MG)$  connects a vertex of a brick to the corresponding vertex of  $MG$ .

**Lemma 1.** *If  $A$  is a connected subgraph of  $\mathcal{B}^+(MG)$  that spans  $Q$ , then  $A - \{\text{portal edges}\}$  is a connected subgraph of  $G_{in}$  that spans  $Q$ .*

The following theorem, proved in Section 3, is central to the proof of correctness of the spanner construction and the approximation scheme. Indeed, taken together, Lemma 1 and Theorem 3 provide a reduction from the Steiner tree problem on  $G_{in}$  to the Steiner tree problem on  $\mathcal{B}^+(MG)$ .

**Theorem 3 (Structure Theorem).** *There exists a constant  $\theta(\epsilon)$  depending polynomially on  $1/\epsilon$  such that, for any choice of portals satisfying the Coverage Property, the corresponding portal-connected graph  $\mathcal{B}^+(MG)$  satisfies*

$$OPT(\mathcal{B}^+(MG), Q) \leq (1 + c\epsilon)OPT(G_{in}, Q)$$

where  $c$  is an absolute constant.

## 2.5 Parcels

First we further decompose  $MG$  into subgraphs called *parcels*, using an integer parameter  $\eta(\epsilon) = \epsilon^{-2}$ .

*Step 4(a):* Do breadth-first search in the planar dual  $MG^*$  starting from  $r$ .

Define the *level* of a vertex of  $MG^*$  (face of  $MG$ ) as its distance from  $r$ . Let  $E_i$  denote the set of edges whose two endpoints are at levels  $i$  and  $i + 1$ .

*Step 4(b):* For  $k = 0, 1, \dots, \eta - 1$ , let  $\mathcal{E}_k = E_k \cup E_{k+\eta} \cup E_{k+2\eta} \cup \dots$ . Let  $k^*$  be the index that minimizes  $\ell(\mathcal{E}_k)$ .

Let  $\mathcal{Y}$  denote the set of connected components of  $MG^* - \mathcal{E}_{k^*}$ . For each  $Y \in \mathcal{Y}$ , let  $H_Y$  denote the subgraph of  $MG$  consisting of the boundaries of faces in  $V(Y)$ . The set of *parcels* of  $MG$  is  $\mathcal{H} = \{H_Y : Y \in \mathcal{Y}\}$ .

*Step 4(c):* Find the set  $\mathcal{H}$  of parcels of  $MG$ .

**Lemma 2.** *The parcel decomposition has the following two properties:*

Radius Property: *The planar dual of each parcel has a spanning tree of depth at most  $\eta + 1$ .*

Boundary-Length Property: *The sum of the lengths of the boundaries of the parcels is at most  $2\ell(MG)/\eta$ .*

## 2.6 New terminals

The next step is to select the new terminals. These new terminals will ensure that the Steiner trees we find later will combine to form a connected subgraph. The parcel-boundary length property ensures that connecting to these new terminals does not increase the lengths of the optimal parcel solutions by much.

*Step 5:* For each parcel  $H$  and for each connected component  $C$  of the boundary of  $H$ , if  $\mathcal{B}^+(MG) - V(C)$  disconnects some terminals, then designate a vertex of  $C$  as a new terminal.

Note that the new terminals are vertices of the mortar graph, not of the bricks. We omit the  $O(n)$  implementation of Step 5.

**Lemma 3.** *The new terminals have the following two properties:*

Spannable Property: *Let  $T$  be a tree in  $\mathcal{B}^+(MG)$  that spans the original terminals and let  $H$  be a parcel. Then  $T \cup \{\text{parcel boundary edges}\}$  contains a tree in  $\mathcal{B}^+(H)$  that spans the original and new terminals in  $H$ .*

Connecting Property: *For each parcel  $H$  that contains a terminal, let  $T_H$  be a tree in  $\mathcal{B}^+(H)$  spanning the original and new terminals belonging to  $H$ . Then  $\bigcup_H T_H$  is a connected subgraph of  $\mathcal{B}^+(MG)$ .*

## 2.7 Optimal solution within the parcels

*Step 6:* For each parcel  $H$ , if  $H$  contains an original or new terminal then find an optimal Steiner tree in  $\mathcal{B}^+(H)$  spanning the original and new terminals in  $H$ .

This step is solved by a  $O(c^{\theta\eta m})$ -time dynamic programming algorithm where  $m$  is the number of edges in  $H$  and  $c$  is a constant. We briefly sketch the idea. Lemma 2 states that the planar dual of  $H$  has a spanning tree  $T^*$  of depth at most  $\eta+1$ . When we apply  $\mathcal{B}^+$  to  $H$  (inserting the bricks), we connect each brick to the corresponding face boundary using at most  $\theta$  portal edges. Suppose we then contract the bricks in  $\mathcal{B}^+(H)$ , turning them into *brick vertices* as shown in Figure 1(e). Each brick vertex is connected to  $MG$  by at most  $\theta$  portal edges. In the dual, these portal edges form a cycle encircling the brick vertex. Add to  $T^*$  all these edges except the one that in the primal graph is incident to an internal brick edge. Let  $\widehat{T}^*$  be the resulting spanning tree. Its depth is at most  $\theta(\eta+1)$ .

Let  $\widehat{T}$  be the set of edges in the contracted graph that do not belong to  $\widehat{T}^*$ . A classical result in planarity states that the complement of a spanning tree of the dual is a spanning tree of the primal, so  $\widehat{T}$  is a spanning tree of the primal. The (primal) input graph had degree three. For each brick,  $\widehat{T}$  has one edge connecting the brick vertex to a vertex  $v$  of the mortar graph. In the input graph there was an edge incident to  $v$  that is no longer present in the contracted graphs, so  $\widehat{T}$  has degree at most three.

Root  $\widehat{T}$  at a non-brick-vertex of degree at most two, and use the rooted tree to guide a dynamic-programming algorithm for Steiner tree in  $\mathcal{B}^+(H)$ . For each vertex  $v$  of  $\widehat{T}$ , the subtree rooted at  $v$  corresponds to a subgraph of  $\mathcal{B}^+(H)$  (replace each brick vertex by the corresponding brick). We show that this subgraph connects with the rest of  $\mathcal{B}^+(H)$  via few edges. Suppose  $v$  is not the root, and let  $e$  be the edge connecting  $v$  to its parent. In the dual,  $e$  is not an edge of  $\widehat{T}^*$ , so it forms a cycle with the simple path in  $\widehat{T}^*$  between its endpoints. Since  $\widehat{T}^*$  has depth at most  $\theta(\eta+1)$ , the cycle has at most  $2\theta(\eta+1)+1$  edges. This shows that, in the primal, the subgraph connects to the rest of  $\mathcal{B}^+(H)$  via at most  $2\theta(\eta+1)+1$  edges, which enables us to do dynamic programming. Each vertex corresponds to a subproblem. The size of the table for this subproblem is  $d^{2\theta(\eta+1)+1}$  where  $d$  is a constant. Because each vertex of  $\widehat{T}$  has at most two children, only two subproblems need to be combined at a time. If  $v$  is a brick vertex, then  $v$  is a leaf in  $\widehat{T}$ , and the subproblem corresponding to  $v$  can be solved using the algorithm of Erickson et al. (Theorem 2).

*Step 7:* Take the union of the edge-sets of all the Steiner trees found in Step 6 (not including portal edges), together with the edges of  $S$ , and return the connected component containing the terminals.

This completes the description of the approximation scheme. Lemma 3 shows that the output is a feasible solution. Lemmas 2 and 3, together with the definition of  $\eta$ , show that the length of the output solution is at most  $(1 +$

$d\epsilon)\text{OPT}(\mathcal{B}^+(MG), Q)$ , and is therefore (by Theorem 3) at most  $(1 + d\epsilon)(1 + c\epsilon)\text{OPT}(G_{in}, Q)$ .

### 3 Proof of the Structure Theorem (Theorem 3)

The construction of what we here call a brick decomposition was given in [6]. Step (i) of the construction involved *cutting open* the input graph along a 2-approximate Steiner tree, obtaining a graph  $G_1$ . Step (ii) used shortest paths to decompose  $G_1$  into “strips.” Step (iii) found some shortest paths within each strip, and Step (iv) designated some of the shortest paths found in Step (iii) as *supercolumns*. For this paper, we define the *mortar graph*  $MG$  of  $G_{in}$  to be the planar embedded subgraph consisting of (A) the edges of the 2-approximate Steiner tree of Step (i), (B) the edges of the shortest paths used in Step (ii), and (C) the edges of the supercolumns. The choice of supercolumns in Step (iv) involves a parameter  $\kappa$ ; choosing  $\kappa(\epsilon) = 8\epsilon^{-2}(1 + \epsilon^{-1})$  yields Lemma 5 (below). In [6], we showed the following properties:

**Lemma 4.** *The boundary of a brick  $B$ , in counterclockwise order, is the concatenation of four paths  $W_B \cup S_B \cup E_B \cup N_B$  such that:*

1. *Every terminal of  $Q$  that is in  $B$  is on  $N_B$  or on  $S_B$ .*
2.  *$N_B$  and  $S_B$  are  $\epsilon$ -short.*
3.  *$V(S_B)$  has a  $\kappa$ -element subset  $\{s_0, \dots, s_\kappa\}$  (in left-to-right order) where for any  $i$  and any vertex  $x \in S_B[s_i, s_{i+1})$ ,  $\text{dist}_{S_B}(x, s_i) < \epsilon \text{dist}_B(x, N_B)$ .*

**Lemma 5.** *Summing over all bricks  $B$ ,  $\sum_B \ell(E_B) + \ell(W_B) \leq \epsilon \text{OPT}(G_{in}, Q)$ .*

#### 3.1 Structural Property of Bricks

This decomposition into bricks is useful because there exists a near-optimal Steiner tree that crosses the boundary of each face of  $MG$  a small number of times. For  $H$  a subgraph of a graph  $G$  and  $P$  a path in  $H$ , a *joining vertex* of  $H$  with  $P$  is a vertex of  $P$  that is the endpoint of an edge of  $H - P$ .

**Theorem 4 (Structural Property of Bricks).** *Let  $B$  be a plane graph with boundary  $N \cup E \cup S \cup W$  and satisfying the brick properties of Lemma 4. Let  $F$  be a set of edges of  $B$ . There is a forest  $\tilde{F}$  of  $B$  with the following properties:*

1. *If two vertices of  $N \cup S$  are connected in  $F$  then they are connected in  $\tilde{F}$ .*
2. *The number of joining vertices of  $F$  with both  $N$  and  $S$  is at most  $\alpha(\epsilon)$ .*
3.  *$\ell(\tilde{F}) \leq (1 + c\epsilon)\ell(F)$ .*

*In the above,  $\alpha(\epsilon) = o(\epsilon^{-5.5})$  and  $c$  is a fixed constant.*

“N”, “E”, “S”, and “W” stand for north, east, south, and west.

In [6], the analogous theorem appeared as Theorem 3.1 with  $\alpha(\epsilon) = 2^{\text{poly}(1/\epsilon)}$  instead. In order to prove Theorem 4, we use the following lemmas.

**Lemma 6.** *Let  $G$  be a planar embedded graph, let  $P$  be an  $\epsilon$ -short path that is a subpath of the boundary  $\partial G$  of  $G$ , and let  $T$  be a tree in  $G$  rooted at a vertex  $r$  whose leaves are exactly  $V(T) \cap V(P)$ . There is another tree  $\widehat{T}$  rooted at  $r$  spanning  $V(T) \cap V(P)$  whose total length is at most  $(1 + c_1 \cdot \epsilon)\ell(T)$  such that  $\widehat{T}$  has at most  $c_2 \cdot \epsilon^{-1.45}$  joining vertices with  $P$ .*

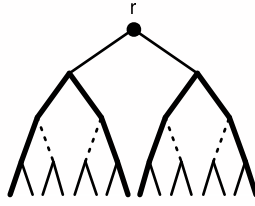
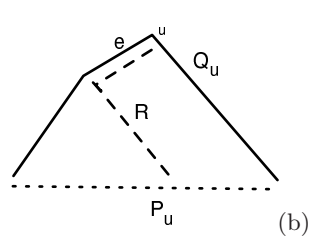
*Proof.* Following [6], as long as there is a vertex  $u$  with at least 3 children, do this: choosing  $u$  to be closest to the root, replace the subtree rooted at  $u$  with (1) the minimal subpath  $P'$  of  $P$  containing all leaves of that subtree, and (2) the shortest  $u$ -to- $P'$  path. The resulting tree  $T'$  has  $\ell(T') \leq (1 + \epsilon)\ell(T)$ .

Define the *level* of a vertex  $v$  to be the number of degree-3 vertices on the  $r$ -to- $v$  path of  $T'$ . Let  $\mathcal{U}$  be the set of degree-3 vertices having level  $k$  (we will choose  $k$  later). A *super-edge* is a maximal descending path in  $T'$  whose internal vertices have degree 2, and its *level* is the level of its first vertex.

For each  $u \in \mathcal{U}$ , replace the subtree  $T'_u$  of  $T'$  rooted at  $u$  with another tree  $T''_u$  rooted at  $u$  that is the union of the shortest subpath  $P_u$  of  $P$  spanning the vertices of  $T'_u \cap P$  and the shortest  $u$ -to- $P_u$  path (Figure 3.1(a)). Let  $T''$  be the result. Analysis is as follows.

For a degree-3 vertex  $u$ , let  $Q_u$  be the path in  $T'$  between  $u$ 's leftmost and rightmost descendent leaves. For each  $i$ , let  $E_i = \bigcup\{Q_u : u \text{ has level } i\}$ , and let  $L_i = \bigcup\{\text{level-}i \text{ super-edges}\} - E_{i-1}$ . See Figure 3.1 (b). Let  $S_i = \bigcup_{j=i}^{\infty} L_j$ .

(a)



**Fig. 3.** (a) Solid line is  $Q_u$ , dashed line is  $R$ , dotted line is  $P_u$ . Root of tree shown is  $u$ , and left child edge is  $e$ . (b) Bold edges:  $E_1$ , dotted edges:  $L_2$ .

Let  $k'$  be the first level  $i$  for which  $\ell(L_i) \leq \ell(S_{i+2})$  (if there is no such level, let  $k' = \infty$ ). Let  $k = \min(k', \lceil \log_{\Phi}(\sqrt{5}(1/\epsilon - 1)) \rceil)$ , where  $\Phi$  is the golden ratio. Since  $k \leq \lceil \log_{\Phi}(\sqrt{5}(1/\epsilon + 1)) \rceil$ , the number of level- $k$  vertices is  $\leq 2^k \leq 11 \cdot \epsilon^{-1.45}$  (for  $\epsilon < 1$ ), which leads to the bound on joining vertices. It remains to show that  $\ell(T'') \leq (1 + \epsilon)\ell(T')$ .

Let  $u$  be a vertex in level  $k$ . Let  $e$  be the unique super-edge in  $L_k$  whose parent is  $u$  (as illustrated in Figure 3.1 (a)). Let  $R$  be the path from  $u$  to  $P$  that traverses  $e$  and subsequently uses only edges of  $E_{k+1} - E_k$ .

$$\ell(T''_u) = \ell(P_u) + \text{dist}_G(u, P_u) \leq (1 + \epsilon)\ell(Q_u) + \ell(R) \quad (1)$$

**Case 1:**  $k = k'$ .  $S_{k+2}$  is disjoint from  $Q_u$  and  $R$ , so the RHS of (1) is at most  $(1 + \epsilon)[\ell(T'_u) + \ell(e) - \ell(S_{k+2} \cap T'_u)] < (1 + \epsilon)[\ell(T'_u) + \ell(L_k \cap T'_u) - \ell(S_{k+2} \cap T'_u)]$ .

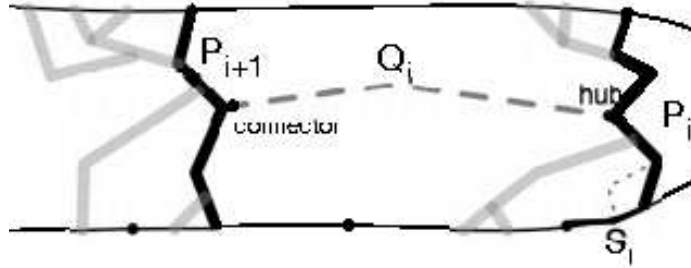
Summing over all level- $i$  vertices  $u$ ,  $\ell(T'') < (1 + \epsilon)[\ell(T') + \ell(L_k) - \ell(S_{k+2})] < (1 + \epsilon)\ell(T')$  since  $\ell(L_k) \leq \ell(S_{k+2})$ .

**Case 2:**  $k \neq k'$ . The RHS of (1) is  $\leq (1 + \epsilon)\ell(Q_u \cup R) + \ell(e) \leq (1 + \epsilon)\ell(T'_u) + \ell(e)$ . Summing over all  $u \in \mathcal{U}$ ,  $\ell(T'') \leq (1 + \epsilon)\ell(T') + \ell(L_k) \leq (1 + \epsilon)\ell(T') + \ell(S_k)$ . Note that  $S_i$  is the disjoint union of  $L_i$  and  $S_{i+1}$ , so  $\ell(S_i) = \ell(L_i) + \ell(S_{i+1})$ . Since  $\ell(L_i) > \ell(S_{i+2})$  for every  $i \leq k$ , we have  $\ell(S_i) \geq \ell(S_{i+2}) + \ell(S_{i+1})$ . It follows that  $\ell(S_1) \geq k^{\text{th}}$  Fibonacci number  $\cdot \ell(S_k)$ . Then by choice of  $k$ ,  $\ell(S_k) \leq \epsilon \ell(S_1) \leq \epsilon \ell(T')$ .  $\square$

**Lemma 7.** *Let  $G$  be a planar embedded graph and let  $T$  be a tree in  $G$  with leaves on an  $\epsilon$ -short path  $P$  that is a subpath of the boundary  $\partial G$  of  $G$ . Let  $p$  and  $q$  be two vertices of  $T$ . There is another tree  $\hat{T}$  spanning  $p, q$ , and the vertices of  $T \cap P$  whose total length is at most  $(1 + c_1 \cdot \epsilon)\ell(T)$  such that  $\hat{T}$  has at most  $c_2 \cdot \epsilon^{-2.45}$  joining vertices with  $P$ .*

The proof for Lemma 7 can be derived from the section titled *Achieving the Third Property* in [6]. It builds on Lemma 6.

*Proof idea for Theorem 4.* We use the term *bases* to refer to the vertices  $s_0, \dots, s_k$  of Part 3 of Lemma 4. We select  $S$ -to- $N$  paths  $P_0, P_1, \dots$ , modifying  $F$  as we go, as follows. (Let  $F'$  denote the modified  $F$ .)



Assume inductively that  $P_i$  has been selected, and let  $x_i$  be its first vertex. Let  $S_i$  be the subpath of  $S$  going west from  $x_i$  to the first base encountered. Note that  $\ell(S_i) \leq \epsilon \ell(P_i)$ . We add  $S_i$  to  $F$ , possibly creating cycles. To fix this, remove an edge not in  $P_i \cup S_i$  from a cycle until no cycles remain. Next, let  $P_{i+1}$  be the eastmost  $S$ -to- $N$  path in  $F$  that starts from a vertex west of  $x_i$  and that is vertex-disjoint from  $P_i \cup S_i$ . By acyclicity, there is at most one path  $Q_i$  from a vertex of  $P_{i+1}$  to a vertex of  $P_i \cup S_i$ . If there is such a path, designate its first vertex as a *connector* of  $P_{i+1}$  and its last vertex as a *hub* of  $P_i$ . Note that the hub has the following property: in the component of  $F' - Q_i - Q_{i-1}$  containing  $P_i$ , every  $S$ -to- $N$  path goes through the hub. If there is no such path  $Q_i$ , we define  $Q_i = \emptyset$  and arbitrarily select as the hub of  $P_i$  any vertex satisfying that property.

Next, we transform  $F'$ , obtaining another forest  $F''$ . For each  $i$ , consider the component of  $F' - Q_i - Q_{i-1}$  that contains  $P_i$ . Decompose this component into two trees: the southern tree consists of the paths from the hub to vertices in  $S$ , and the northern tree consists of the paths from the hub to vertices in  $N$ . We apply Lemma 6 to whichever of these trees does not contain a connector (taking  $r$ =the hub), and we apply Lemma 7 to whichever does contain a connector (taking  $p$ =the hub and  $q$ =the connector).  $\square$

### 3.2 Completion of the proof of Theorem 3

The Structure Theorem (Theorem 3) states that  $\text{OPT}(\mathcal{B}^+(MG), Q) \leq (1 + c\epsilon)\text{OPT}(G_{in}, Q)$ . We now give the proof using Theorem 4.

*Proof.* We start from an optimal solution  $T$  to the Steiner tree problem in  $G_{in}$  and gradually transform it into a solution  $\hat{T}$  to the Steiner tree problem in  $\mathcal{B}^+(MG)$ , while approximately preserving its length. First, let  $T_1$  be the union of  $T$  with the east and west boundaries ( $E_B$  and  $W_B$ ) for every brick  $B$  in  $G$ . Using Lemma 5, we have  $\ell(T_1) \leq \text{OPT} + \epsilon\text{OPT}(G_{in}, Q)$ . Remove edges (other than east/west boundary edges) to break cycles.

Next, apply Theorem 4 to the subgraph of  $T_1$  that is contained in each brick, obtaining  $T_2$  such that  $\ell(T_2) \leq (1 + c'\epsilon)\ell(T_1)$ .

Next, we must obtain a solution in  $\mathcal{B}^+(MG)$ . Let  $T_2^a$  be the set of brick-boundary edges of  $T_1$ , and let  $T_2^b$  the other edges of  $T_2$ . Let  $T_3$  be the set of edges of  $\mathcal{B}^+(MG)$  consisting of (a) the edges of  $MG$  corresponding to edges of  $T_3^a$ , and (b) the edges of  $T_3^b$ . Note that  $T_3$  is not a connected subgraph of  $\mathcal{B}^+(MG)$ . A path in  $T_2$  might pass from the interior of a brick to the boundary; the corresponding sequence of edges in  $T_3$  would have a gap: the “path” would stop at a vertex of the brick boundary, and resume at the corresponding vertex of the mortar boundary. To close the gap, we must add paths connecting each to the nearest portal vertex associated with that brick and then add the corresponding portal edge. The resulting graph  $T_4$  is connected.

It remains to bound the length of all these detours. For brick  $B$ , the distance to the nearest portal is at most  $\ell(\partial B)/\theta$ , so the length of the detour is at most  $2\ell(\partial B)/\theta$ . By Theorem 4, the number of detours for this brick is at most  $\alpha$ , so the length of all these detours is at most  $2\alpha\ell(\partial B)/\theta$ . Summing over all bricks and using the bound from Section 2.1 on the length of the mortar graph, we obtain a bound of  $10\alpha\epsilon^{-1} \cdot \text{OPT}(G_{in}, Q)/\theta$ . The choice of  $\theta$  ensures that this is at most  $\epsilon \cdot \text{opt}(G_{in}, Q)/\theta$ .  $\square$

## References

1. S. Arora. Polynomial-time approximation schemes for euclidean TSP and other geometric problems. *JACM*, 45(5):753–782, 1998.
2. P. Berman and V. Ramaiyer. Improved approximations for the Steiner tree problem. *J. Alg.*, 17:381–408, 1994.

3. M. Bern. Faster exact algorithms for Steiner trees in planar networks. *Networks*, 20:109–120, 1990.
4. M. Bern and D. Bienstock. Polynomially solvable special cases of the Steiner problem in planar networks. *Ann. Op. Res.*, 33:405–418, 1991.
5. M. Bern and P. Plassmann. The Steiner problem with edge lengths 1 and 2. *IPL*, 32:171–176, 1989.
6. G. Borradaile, C. Kenyon-Mathieu, and P. Klein. A polynomial-time approximation scheme for Steiner tree in planar graphs. In *18th SODA*, pages 1285–1294, 2007.
7. R. Erickson, C. Monma, and A. Veinott. Send-and-split method for minimum-concave-cost network flows. *Math. Op. Res.*, 12:634–664, 1987.
8. M. Garey and D. Johnson. The rectilinear Steiner tree problem is NP-complete. *SIAM J. Appl. Math.*, 32(4):826–834, 1977.
9. M. Henzinger, P. Klein, S. Rao, and S. Subramanian. Faster shortest-path algorithms for planar graphs. *J. Comput. System Sci.*, 55(1):3–23, 1997.
10. S. Hougardy and H. J. Prömel. A 1.598 approximation algorithm for the Steiner problem in graphs. In *10th SODA*, pages 448–453, 1999.
11. R. Karp. On the computational complexity of combinatorial problems. *Networks*, 5:45–68, 1975.
12. M. Karpinski and A. Zelikovsky. New approximation algorithms for the Steiner tree problem. *Journal of Combinatorial Optimization*, 1:47–65, 1997.
13. P. Klein. A linear-time approximation scheme for planar weighted TSP. In *46th FOCS*, pages 647–647, 2005.
14. L. Kou, G. Markowsky, , and L. Berman. A fast algorithm for Steiner trees. *Acta Informatica*, 15:141–145, 1981.
15. K. Mehlhorn. Approximation algorithm for the Steiner problem in graphs. *IPL*, 27(3):125–128, 1988.
16. J. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric tsp, k-mst, and related problems. *SIAM J. Comput.*, 28(4):1298–1309, 1999.
17. H. J. Prömel and A. Steger. RNC approximation algorithms for the Steiner problem. In *39th STOC*, pages 559–570, 1997.
18. J. Provan. An approximation scheme for finding Steiner trees with obstacles. *SIAM J. Comput.*, 17(920-934), 1988.
19. J. Provan. Convexity and the Steiner tree problem. *Networks*, 18:55–72, 1988.
20. S. Rao and W. Smith. Approximating geometrical graphs via "spanners" and "banyans". In *30th STOC*, pages 540–550, 1998.
21. G. Robins and A. Zelikovsky. Tighter bounds for graph Steiner tree approximation. *SIAM J. Discret. Math.*, 19(1):122–134, 2005.
22. H. Takahashi and A. Matsuyama. An approximate solution for the Steiner problem in graphs. *Mathematica Japonicae*, 24:571–577, 1980.
23. P. Widmayer. A fast approximation algorithm for Steiner's problem in graphs. In *Graph-Theoretic Concepts in Computer Science*, volume 246 of *LNCS*, pages 17–28. Springer Verlag, 1986.
24. Y. Wu, P. Widmayer, and C. Wong. A faster approximation algorithm for the Steiner problem in graphs. *Acta informatica*, 23(2):223–229, 1986.
25. A. Zelikovsky. Better approximation bounds for the network and Euclidean Steiner tree problems. Technical Report CS-96-06, University of Virginia, 1994.
26. A. Zelikovsky. An 11/6-approximation algorithm for the network Steiner problem. *Algorithmica*, 9:463–470, 1999.