A Polynomial-Time Approximation Scheme for Steiner Tree in Planar Graphs

Glencora Borradaile[*][†]   Claire Kenyon-Mathieu[†]   Philip Klein[‡][†]

Abstract

We give an $O(n \log n)$ approximation scheme for Steiner tree in planar graphs.

1   Introduction

The *Steiner tree problem in networks* is the following. Given a graph with edge lengths, and given a subset S of the vertices, find a minimum-length connected subgraph that spans all vertices in S (and possibly some others). The vertices in S are called *terminals*. The minimum spanning tree problem is the special case where every vertex in the graph is a terminal.

Steiner tree in networks is one of the most well-studied problems in combinatorial optimization. It was one of Karp's original NP-complete problems, and is now known to be max SNP-complete (Bern and Plassman [6]), so there is no polynomial-time approximation scheme for the problem unless $P = NP$. Takahashi and Matsuyama [16] and Kou, Markowsky and Berman [11] presented 2-approximation algorithms. Their running time was improved by Wu, Widmayer, and Wong [18], Widmayer [17], and Mehlhorn [12].

For the case of terminals in a constant-dimensional space and distances given by the Euclidean metric, Arora [1] gave a polynomial-time approximation scheme (the time bound is a polynomial in n, and the degree of the polynomial depends on ϵ), and Rao and Smith [15] gave an $O(n \log n)$ approximation scheme; here n denotes the number of terminals.

A natural generalization of Euclidean metrics in two-dimensional space (and a natural specialization of the class of all metrics) is the family of undirected planar graphs with positive edge lengths. Until now, the best known approximation for the Steiner tree problem in planar graphs was the general 2-approximation. Here we give an approximation scheme.

THEOREM 1.1. *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 $O(n \log n)$, where n is the number of vertices of G.*

The construction uses a dynamic-programming algorithm of Erickson, Monma, and Veinott [7] to find an optimal Steiner tree in the special case where the k terminals all lie on the boundary of a planar embedded graph, in time $O(nk^3 + (n \log n)k^2)$.[1]

Overview Our algorithm uses a paradigm to design approximation schemes in planar graphs with edge-lengths, previously used by Klein [9] in the context of the traveling-salesman problem. The first step of the paradigm involves removing edges to define a subgraph of the input graph that approximately preserves the value of the optimum and that has total length at most a constant times that of the optimum. The idea of using such a spanner-type result for an approximation scheme was also used by Arora, Grigni, Karger, Klein, and Woloszyn [2], and by Rao and Smith [15]. In fact, Rao and Smith gave a version of Theorem 1.2 for fixed-dimensional Euclidean space.[2]

Formally, fix a number $0 < \epsilon < 1$, and for a graph G with edge lengths and a subset S of the vertices of G, let $\mathrm{OPT}_S(G)$ denote the minimum length of a Steiner tree in G that spans S. A subgraph H of G is a *Steiner-tree spanner* with respect to S if it has the following two properties:

(spanning) There is a tree in H that spans S and has length at most $(1 + \epsilon)\mathrm{OPT}_S(G)$, and

(short) The total length of H is at most some function of ϵ times $\mathrm{OPT}_S(G)$.

Much of the paper is devoted to proving the following result.

---

[1]This algorithm has been generalized by Bern [4] and by Bern and Bienstock [5] to handle some additional special cases, e.g. where the terminals lie on a constant number of faces. Provan [14, 13] used the same approach to give exact and approximate algorithms for some geometric special cases.

[2]Their construction was in fact more powerful, in that the subgraph included a nearly optimal Steiner tree spanning *any* subset of S.

THEOREM 1.2. *There is a function $f(\cdot)$ such that, 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 spanner whose total length is at most $f(\epsilon)$ times the length of the optimal Steiner tree in G spanning S. The running time is $O(n \log n)$, where n is the number of vertices of G.*

The function $f(\epsilon)$ and the constant in the running time are doubly exponential in $1/\epsilon$. Consequently, the running time of the resulting approximation scheme is triply exponential in $1/\epsilon$.

We now sketch the construction, detailed in Section 3.2, used to prove Theorem 1.2.

We start with a planar-graph decomposition due to Klein [10] We decompose the planar embedded graph via shortest paths into long, skinny subgraphs called *strips*. We put the strip boundaries in the Steiner-tree spanner. Each strip is cut width-wise by shortest paths which we call *columns* in this paper.

Then we choose a subset of the columns (which we call *supercolumns*) using a *shifting* technique (as in Baker [3]): For some large constant k depending on ϵ, we take every k^{th} column, choosing the phase shift so as to ensure that the total lengths of the supercolumns is at most $1/k$ times the total length of all columns. We put the supercolumns in the Steiner-tree spanner.

Next, for each strip and each pair of consecutive supercolumns within that strip, we focus on the subgraph enclosed by the supercolumns and the strip boundaries. We designate as *portals* [1] a constant number of vertices along the strip boundaries at regular intervals. Finally, for each subset S' of the portals, using the algorithm from [7] we find an optimal Steiner tree of S' in the subgraph and add it to the Steiner-tree spanner.

2 Approximation Scheme

Now we sketch the approximation scheme of Theorem 1.1, using the framework of [9] as it applies to the Steiner tree problem.

Spanner step: Find a Steiner-tree spanner H of G according to Theorem 1.2.

Thinning step: Applying a Baker-type shifting technique [3] to the planar dual of H as in [10], select a set C of edges of H whose total length is at most $1/f(\epsilon)$ times the weight of H (where f is the function given in Theorem 1.2), and such that the graph K obtained from H by contracting edges of C has branch-width at most $2(f(\epsilon) + 1)$.

Dynamic-programming step: Use dynamic programming (see, e.g., [8]) to find the optimal solution in K.

Lifting step: Convert the optimal solution found in the previous step to a solution for H by incorporating some of the edges of C.

Analysis of running time. According to Theorem 1.2, the spanner step takes $O(n \log n)$ time for fixed ϵ. Thinning and lifting take $O(n)$ time. The dynamic-programming step takes time $O(c^{f(\epsilon)}n)$. Thus the overall algorithm takes $O(n \log n)$ time for fixed ϵ.

3 Proof of Theorem 1.2

3.1 Preliminaries For a graph G, let $\mathrm{dist}_G(\cdot, \cdot)$ denote the shortest path distance. We omit the subscript when it is clear which graph is intended.

For a path P, we denote the first vertex of P by $\mathrm{start}(P)$ and the last vertex by $\mathrm{end}(P)$. A vertex of P other than $\mathrm{start}(P)$ and $\mathrm{end}(P)$ is *internal* to P. For vertices x, y on P, we denote by $P[x, y]$ the x-to-y subpath of P. We denote by $P(x, y)$ the path obtained from $P[x, y]$ by omitting the first vertex x and the last vertex y. $P(x, y]$ and $P[x, y)$ are similarly defined.

A subgraph H of a graph G is also considered a subset of the edges of G. The set of vertices of G that are endpoints of edges in H is denoted $V(H)$.

The *boundary* 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 the boundary of H.

For an assignment $\ell(\cdot)$ of lengths to edges and a set A of edges, we use $\ell(A)$ to denote $\sum_{e \in A} \ell(e)$.

3.2 Construction Let G be a planar embedded graph with positive edge-lengths $\ell(\cdot)$, and let S be a set of vertices. The algorithm consists of seven steps. Steps 1 through 3 of the algorithm to build a spanner for (G, S) are identical to the first three steps in Klein's construction [10] of a subset spanner.

Step 1: Cutting the graph open The first step is to find a 2-approximate Steiner tree T spanning S in G. Viewed as a planar embedded graph, the single face of T is an Euler tour that traverses each edge once in each direction. Let G_1 be the planar embedded graph obtained by duplicating each edge of T and introducing multiple copies of vertices, so as to transform this Euler tour into a simple cycle enclosing no vertices and bounding a single face. Change the embedding to take this face to be the infinite face of G_1. This process is illustrated in Figure 1.

LEMMA 3.1. (Klein [10])

$$\ell(\textit{boundary of } G_1) \leq 4\,\mathrm{OPT}_S(G).$$

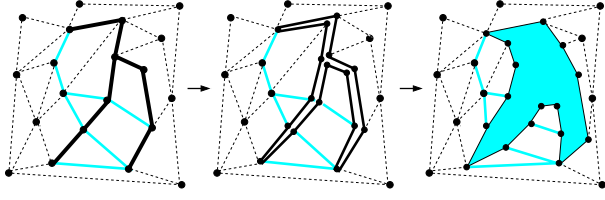This first step can be done in $O(n \log n)$ time [12, 17, 18].

Figure 1: The process of cutting open a graph along a tree: duplicate edges, replicate vertices and creating a new face (the shaded face). For example, the bold edges form a 2-approximate Steiner tree that we cut open. The light solid edges form the minimum-length Steiner tree: in the resulting graph, these edges form a forest.

Step 2: Breaking the graph into strips The second step is to decompose G_1 into *strips*. Let H be the boundary of G_1. Let $H[x, y]$ denotes the unique nonempty counterclockwise x-to-y subpath of H.[3] We use a recursive algorithm.

Find[4] vertices x, y on H such that

- $(1 + \epsilon)\mathrm{dist}_{G_1}(x, y) < \ell(H[x, y])$, and

- $(1+\epsilon)\mathrm{dist}_{G_1}(x', y') \geq \ell(H[x', y'])$ for every x', y' in $H[x, y]$ such that $x' \neq x$ or $y' \neq y$

Let B be a shortest path from x to y in G_1. Then the subgraph enclosed by $H[x, y] \cup B$ is called a *strip*. The path B is called the *blue* boundary of the strip. The path $H[x, y]$ is called the *red* boundary of the strip, and is denoted R. We think of the blue boundary as the upper boundary and the red boundary as the lower boundary. With this orientation, we direct R and B from left to right.

Recursively decompose the subgraph of G_1 enclosed by $B \cup H - H[x, y]$ into strips (if this subgraph is nontrivial).

LEMMA 3.2. Inequality (10), Klein [10] *The total length of all the boundary edges of all the strips is at most $(\epsilon^{-1} + 1)$ times the length of H.*

Klein [10] shows that the strip decomposition of an n-vertex planar graph can be found in $O(n \log n)$ time.

---

[3]If $x = y$ then $H[x, y]$ is defined to be H.

[4]Such a pair of vertices always exists since choosing $x = y$ satisfies the first condition.

Step 3: Finding short paths crossing the strips We have decomposed G_1 into strips. Next, for each strip we find short paths crossing the strip. We call these short paths *columns* in this paper. Consider a strip, and let R and B be its red and blue boundaries.

We select vertices $r_0, r_1, \ldots$ inductively as follows. Let r_0 be the left endpoint common to R and B. For $i = 1, 2, \ldots$, find the vertex r_i on R such that
- $(1+\epsilon)\mathrm{dist}_{G_1}(r_i, B) < \mathrm{dist}_R(r_i, r_{i-1}) + \mathrm{dist}_{G_1}(r_{i-1}, B)$, and
- $(1 + \epsilon)\mathrm{dist}_{G_1}(x, B) \geq \mathrm{dist}_R(x, r_{i-1}) + \mathrm{dist}_{G_1}(r_{i-1}, B)$ for every x in $R[r_{i-1}, r_i]$.

For $i = 0, 1, 2, \ldots$, column C_i is defined to be a shortest path from r_i to B. Note that C_0 is a path with no edges since r_0 belongs to B. We also include as a column the no-edge path starting and ending at the rightmost vertex common to R and B.
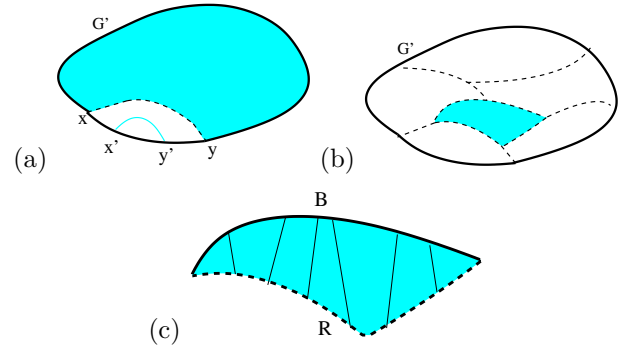

Figure 2: (a) The first strip is created by a path (dashed) connecting x to y. The distance between every pair of vertices, x' and y', between x and y on the boundary is well approximated by the boundary distance. We recurse on the shaded face. (b) A graph is divided into strips (by the dashed lines). One strip is shaded and enlarged in (c). Columns (vertical lines) are taken from the set of shortest paths from the lower, red boundary R (dashed) to the upper, blue boundary B (solid).

LEMMA 3.3. Lemma 5.2, Klein [10] *The sum of the lengths of the columns in a strip is at most $\epsilon^{-1}\ell(R)$.*

It is easy to find all the columns in $O(n \log n)$ time.

Step 4: Selecting supercolumns, dividing each strip into panels Let $k = 16\epsilon^{-2}(1 + \epsilon^{-1})$. For each strip, we select a subset of the columns of that strip as follows. Let $C_0, C_1, \ldots, C_s$ be the columns. For $i = 0, 1, \ldots, k - 1$, let $\mathcal{C}_i = \{C_j : j \equiv i \pmod{k}\}$. Let $i^* = \min \arg_i \ell(\mathcal{C}_i)$. We designate the columns in $\mathcal{C}_{i^*}$ as *supercolumns*.

LEMMA 3.4. *The sum of lengths of the supercolumns in a strips is at most $1/k$ times the sum of the lengths of the columns in the strip.*

We have decomposed G_1 into strips. The supercolumns of a strip further decompose that strip into subgraphs called *panels*. Namely, for each pair of consecutive supercolumns of a strip, the subgraph of the strip bounded by those supercolumns is a panel. The boundary of a panel P consists of the left supercolumn $C_1(P)$, the right supercolumn $C_2(P)$, a subpath $R(P)$ of the red boundary of the strip, and a subpath $B(P)$ of the blue boundary of the strip. We refer to these pieces of the boundary as C_1, C_2, R, B when the choice of the panel P is clear.

Each panel P possesses the following *panel properties*:

1. For every pair x, y of vertices in B, $\mathrm{dist}_B(x, y) < (1 + \epsilon)\mathrm{dist}_P(x, y)$.

2. For every pair x, y of vertices in R, $\mathrm{dist}_R(x, y) < (1 + \epsilon)\mathrm{dist}_P(x, y)$.

3. There exist vertices $r_0, r_1, r_2, \ldots, r_s$ on R such that $r \leq k$ and, for every $1 \leq i < s$, for every vertex x on R, for $i^* = \max\{i \ : \ r_i$ is left of x in $R\}$, we have

$$\mathrm{dist}_R(x, r_{i^*}) + \mathrm{dist}_P(r_{i^*}, B) \leq (1 + \epsilon)\mathrm{dist}_P(x, B).$$

Step 5: Selecting portals For each panel P, we designate some of the boundary vertices as *portals*. Let $h = h(\epsilon)$ be a function to be determined later, and let $\tau(P) = \frac{\ell(R(P)) + \ell(B(P))}{h}$.

We select portals among the vertices of R inductively as follows. The first portal is the leftmost vertex v_{left} of R. Inductively, let w be the vertex on R immediately to the right of the last portal designated, and designate as the next portal the rightmost vertex v of R such that $\ell(R[w, v]) \leq \tau(P)$. Finally, the rightmost vertex v_{right} of R is also designated as a portal.

For each portal v except for v_{left} and v_{right}, the length of the subpath of R from the previous portal to v is more than $\tau(P)$. Hence the number of portals is less than $2 + \ell(R)/\tau(P)$.

We similarly define portals on B. The total number of portals on R and B is less than $4 + (\ell(R) + \ell(B))/\tau(P)$. We obtain the following *portal properties*:

- *representative:* For any vertex u on R, there is a portal v such that $\ell(R[u, v]) \leq \tau(P)$. For any vertex u on B, there is a portal v such that $\ell(B[u, v]) \leq \tau(P)$.

- *cardinality:* The number of portals on the boundary of the panel is at most $h + 4$.

Step 6: Finding Steiner trees For each panel P and for each subset S of portals on that panel, find an optimal S-spanning Steiner tree in the panel using the algorithm of Erickson et al. [7] mentioned in the introduction, where the lengths of edges of $C_1(P)$ and $C_2(P)$ are reassigned zero. Since the number of terminals of each Steiner-tree problem solved here is a constant, the time for this sixth step is $O(n \log n)$.

Step 7: Putting it together The spanner H is defined as the union of the following: (1) the edges of the approximate Steiner tree T, (2) the edges of all strip boundaries, (3) the edges of all supercolumns, and (4) the edges of all Steiner trees found in Step 6.

The running time of the construction is $O(n \log n)$.

3.3 The shortness property

LEMMA 3.5. *The total length of edges in all strip boundaries is at most $4(\epsilon^{-1} + 1)\,\mathrm{OPT}_S(G)$.*

Proof. Combine lemmas 3.1 and 3.2. □

The length of each Steiner tree constructed in Step 6 is at most $\ell(R) + \ell(B)$. There are at most $h + 4$ portals, so the number of Steiner trees is at most 2^{h+4}. Thus the total length of all edges in these Steiner trees is at most $2^{h+4}(\ell(R) + \ell(B))$. Summing over panels and using Lemma 3.5 and the fact that each strip boundary edge occurs in at most two strips gives a bound of at most $2^{h+4}8(1 + \epsilon^{-1})\mathrm{OPT}_S(G)$.

Together, the total length of the Steiner-tree spanner is at most $\mathrm{OPT}_S(G)$ times $4(1 + \epsilon^{-1}) + \epsilon + 2^{h+7}(1 + \epsilon^{-1})$. This proves the *short* property of Theorem 1.2.

The remainder of the paper is devoted to proving the *spanning* property of the Steiner-tree spanner.

3.4 The spanning property.

LEMMA 3.6. *The sum over all panels of the lengths of all the supercolumns is at most $\frac{\epsilon}{4}\,\mathrm{OPT}_S(G)$.*

Proof. Combine lemmas 3.1, 3.2, 3.3, 3.4 and definition of k. □

The key to proving the spanning property is the following theorem, proved in Section 4.

THEOREM 3.1. *Let P be a plane graph satisfying the panel properties, with boundary $C_1 \cup C_2 \cup R \cup B$. Let F be a set of edges strictly enclosed by the boundary of P. There is a forest $\tilde{F}$ of P with the following properties:*

1. *If two vertices of $V(R \cup B)$ are connected in F then they are connected in $\tilde{F}$.*

2. *The number of vertices in $V(R \cup B)$ that are endpoints of edges of $\tilde{F} - (R \cup B)$ is at most $a(\epsilon)$.*

3. $\ell(\tilde{F}) \leq (1 + c\epsilon)\ell(F)$

Here c is an absolute constant and $a(\epsilon)$ is a certain function of ϵ.

Now we show how to use Theorem 3.1 to prove the spanning property.

Let T^* be an optimal Steiner tree in G spanning S. Considering Step 1 of the spanner construction, map each edge of T^* to an edge in G_1, choosing one of the two duplicate edges arbitrarily when the edge of T^* happens to also be an edge of T. Let F^* be the resulting set of edges of G_1.

Consider Step 4 of the spanner construction. For each panel P, let $F(P)$ denote the set of edges of F^* that are strictly enclosed by the boundary of P together with the edges of $C_1(P)$ and $C_2(P)$. Apply Theorem 3.1 with $\bar{\epsilon} = \epsilon/3c$ to obtain a forest $\tilde{F}(P)$ such that $\ell(\tilde{F}(P)) \leq (1 + \frac{\epsilon}{3})\ell(F(P))$.

Let $Q(P)$ denote the set of vertices of $V(R(P) \cup B(P))$ that are endpoints of edges of $\tilde{F}(P) - (R(P) \cup B(P))$. Note that $|Q(P)| \leq a(\epsilon)$. For each vertex $q \in Q(P)$, let $\rho(q)$ denote the subpath of R or B (whichever contains q) connecting q to the nearest portal. Note that $\ell(\rho(q)) \leq \tau(P)$, so we have

$$
\begin{aligned}
\sum_{q \in Q(P)} \ell(\rho(q)) &\leq |Q(P)|\tau(P) \\
&\leq a(\epsilon) \cdot \frac{\ell(R(P)) + \ell(B(P))}{h(\epsilon)} \\
&\leq \frac{d(P)}{2g(\epsilon)}
\end{aligned}
$$

where we define $h(\epsilon) = 2g(\epsilon) \cdot a(\epsilon)$ and $d(P) = \ell(R(P)) + \ell(B(P))$.

Let $F_1(P)$ denote $\tilde{F}(P) \cup \bigcup\{\rho(q) : q \in Q(P)\}$. We have

$$
\begin{aligned}
\ell(F_1(P)) &\leq \ell(\tilde{F}(P)) + \sum_{q \in Q(P)} \ell(\rho(q)) \\
&\leq (1 + \epsilon/3)\ell(F(P)) + \frac{d(P)}{2g(\epsilon)}
\end{aligned}
$$

For each connected component C of $F_1(P)$, there is an optimal Steiner tree $F_C(P)$ found in Step 6 that connects the portals in $V(C)$. By construction, the edges of $F_C(P)$ lie in the spanner. Let $F_2(P)$ denote the union of these Steiner trees. By optimality of the Steiner trees,

$$
\begin{aligned}
\ell(F_2(P)) &\leq \sum_C \ell(C) \\
&\leq \ell(F_1(P))
\end{aligned}
$$

Let $F_3(P) = F_2(P) \cup \bigcup\{\rho(q) : q \in Q(P)\}$. Since the paths $\rho(q)$ lie in $R(P)$ and $B(P)$, all the edges of $F_3(P)$ lie in the spanner. We have

$$
\ell(F_3(P)) \leq (1 + \epsilon/3)\ell(F(P)) + \frac{d(P)}{g(\epsilon)}
$$

For a vertex x internal to one of the supercolumns $C_1(P)$ and $C_2(P)$, define x_P to be the red vertex of that supercolumn. If x is in $V(R(P) \cup B(P))$, on the other hand, define $x_P = x$. Note that in either case $x_P \in V(R(P) \cup B(P))$.

Let $F_4(P) = F_3(P) \cup (F^* \cap (R(P) \cup B(P)))$.

CLAIM. *Suppose x and y are vertices of $V(C_1(P) \cup C_2(P) \cup R(P) \cup B(P))$ that are connected in $F^* \cap P$. Then x_P and y_P are connected in $F_4(P)$.*

Proof. (For notational simplicity, we omit "(P)" in the proof.) Since F_4 includes the edges of $F^* \cap (R \cup B)$, it suffices to prove the claim for the case where the x-to-y path in $F^* \cap P$ contains no edges of $R \cup B$. Because F contains C_1 and C_2, there is an x_P-to-y_P path in F that contains no edges of $R \cup B$. By Theorem 3.1, there is an x_P-to-y_P path in $\tilde{F}$. Combining this path with $\rho(x_P)$ and $\rho(y_P)$ yields a path in F_1 between the portal closest to x_P and the portal closest to y_P, and hence a path in F_2 with the same endpoints. Combining this path with $\rho(x_P)$ and $\rho(y_P)$ yields an x_P-to-y_P path in F_3. $\square$

Now let $\tilde{T} = \bigcup_P F_4(P)$ where the union is over all panels.

CLAIM. *The subgraph of G consisting of edges of $\tilde{T}$ connects every pair of vertices of S.*

Proof. Consider a pair of vertices of S. There is a path in the optimal Steiner tree T^* between them. This path decomposes into minimal paths between the vertices of the 2-approximate Steiner tree T. Each such path in turn is a path D in G_1 between boundary vertices of G_1. The path D in G_1 decomposes into subpaths where each subpath lies entirely within a panel P. Consider such a subpath, and let x and y be its start and end vertices. Then there is a corresponding x_P-to-y_P path in $F_4(P)$. We show that combining these paths yields a path in $\tilde{T}$ with the same endpoints as D.

Consider two consecutive subpaths β_1 and β_2 of D, one in panel P_1 and one in panel P_2. Let γ_1 and γ_2 be the corresponding paths in $F_4(P_1)$ and $F_4(P_2)$. Let x be the vertex common to the two paths, i.e. $x = \text{end}(\beta_1) = \text{start}(\beta_2)$. If x is internal to a column, then x_{P_1} is the red vertex of that column, and the same is true for x_{P_2}, so $x_{P_1} = x_{P_2}$. This shows that γ_1 concatenated with γ_2 is a path.

Now consider the first subpath β of D, and the panel P in which it lies. The first vertex $x = \text{start}(\beta)$ belongs to the boundary of G_1 and hence cannot be an internal vertex of a column, so $x_P = x$. The same argument applies to the last vertex of the last subpath of D. This proves that the concatenation of the paths in $F_4(P)$ is a path with the same start and end vertices as D. $\quad\square$

We have $\ell(F^*) = \text{OPT}_S(G)$. Write $F^* = F_1^* \cup F_2^*$, where we define F_1^* to be the set of edges of F^* internal to strips, and we define F_2^* to be the set of edges of F^* belonging to strip boundaries. It remains to bound the length of $\tilde{T}$.

$$
\begin{aligned}
&\ell(\tilde{T}) \\
&= \ell\Big(\bigcup_P F_4(P)\Big) \\
&= \ell\Big(\bigcup_P F_3(P) \cup (F^* \cap (R(P) \cup B(P)))\Big) \\
&= \ell\Big(\bigcup_P F_3(P)\Big) + \ell\Big(\bigcup_P F^* \cap (R(P) \cup B(P))\Big) \\
&\le \sum_P \Big((1 + \tfrac{\epsilon}{3})\ell(F(P)) + \tfrac{d(P)}{g}\Big) + \ell(F_2^*)
\end{aligned}
$$

by our upper bound on $\ell(F_3(P))$.

Next we bound each of the two sums separately.

$$
\begin{aligned}
&\sum_P \ell(F(P)) \\
&\le \ell(F_1^*) + \sum_P (\ell(C_1(P)) + \ell(C_2(P))) \\
&\le \ell(F_1^*) + \tfrac{\epsilon}{4}\text{OPT}_S(G)
\end{aligned}
$$

by Lemma 3.6. Finally,

$$
\begin{aligned}
\sum_P \frac{d(P)}{g} &\le \frac{1}{g}8(\epsilon^{-1} + 1)\text{OPT}_S(G) \\
&\le \frac{\epsilon}{3}\text{OPT}_S(G)
\end{aligned}
$$

by Lemma 3.2 and the fact that each strip-boundary edge belongs to at most two panels, and where we define $g(\epsilon) = 24(\epsilon^{-1} + 1)$.

Combining these inequalities and using $\epsilon < 1$, we obtain $\ell(\tilde{T}) \le (1 + \epsilon)\text{OPT}_S(G)$, proving the spanning property.

4 Proof of the structure theorem

In this section we prove Theorem 3.1. We consider a planar embedded graph graph G bounded by R, C_1,

B, and C_2 and satisfying the panel properties, and a forest F_0. We show how to modify F_0 to get a forest that preserves the connectivity of F_0 among vertices in in $R \cup B$ and has length at most $1 + c\epsilon$ times the total length of F_0. For this proof, we refer to the vertices in $R \cup B$ as *terminals*.

A vertex of B is designated *blue*, and a vertex of R not in B is designated *red*. Let T be a tree in G. An edge e of T is a *spine edge* if the two connected components of $T - \{e\}$ both contain both blue vertices and red vertices.

LEMMA 4.1. *For each tree T, the spine edges of T form a simple path.*

We call this simple path a *spine*.

A vertex is a *spine vertex* if it is the endpoint of a spine edge. If a bichromatic tree T contains no spine edges, designate as the spine vertex some vertex v such that each component of $T - \{v\}$ is monochromatic.

For a tree T, define a *good path* to be a simple path in T containing at least one edge and containing no spine edges and no edges of $R \cup B$. A spine vertex v is *red-attached* if there is a good v-to-red path. If v is red-attached, the *v-rooted off-spine red subtree* is the tree formed by the union of all the good v-to-red paths. Note that all the leaves of an off-spine red subtree are red vertices. *Blue-attached* and *off-spine blue subtree* are defined similarly.

A *run* is a maximal subpath P of a spine such that at least one of the following properties holds:

1. no internal vertex of P is red-attached, or

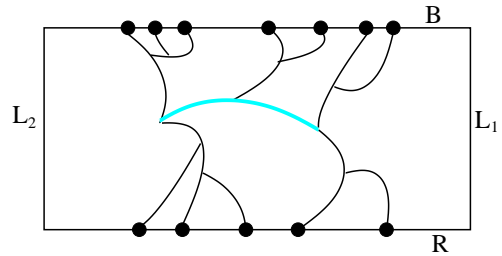2. no internal vertex of P is blue-attached.

Figure 3: Here is a tree spanning blue vertices on B and red vertices on R in a panel. Each edge of the shaded path is a spine edge.

We carry out the modifications to F_0 in four stages, producing, in turn, F_1, F_2, F_3, and F_4, with the following properties.

1. For $i = 1, 2, 3$, each tree T of $F_i - R \cup B$ contains both red and blue vertices.

2. F_2 has at most $c_1\epsilon^{-3}$ runs.

3. For $i = 3, 4$, the number of off-spine subtrees in F_i is at most $\epsilon^{-1} + 6$ times the number of runs in F_2.

4. Each off-spine subtree of F_4 has at most $c_3^{1/\epsilon}$ leaves.

In the above c_1, c_2, and c_3 are small absolute constants.

Let $\tilde{F} = F_4$. In carrying out the modifications, we only remove an edge if the edge belongs to a cycle. Hence if two vertices of $V(R \cup B)$ are connected in F, they are connected in $\tilde{F}$. By combining Properties 2-4, we infer that the number of leaves of off-spine trees is at most $c_1(\epsilon^{-4} + \epsilon^{-3})c_3^{1/\epsilon}$. Property 1 implies that every terminal that is an endpoint of an edge of $\tilde{F} - (R \cup B)$ is a leaf of an off-spine tree. Finally, we also show that $\ell(\tilde{F}) \leq (1 + c\epsilon)\ell(F)$ for some constant c. Thus $\tilde{F}$ fulfills the requirements of Theorem 3.1.

Achieving the first property

Let T be a monochromatic tree of $F_0 - R \cup B$ and suppose without loss of generality that T has red vertices but no blue vertices. Let x and y be the left and rightmost vertices of T on R. Then T must have length at least $\text{dist}(x, y)$. Since R is an approximately short path, $\ell(R[x, y]) < (1 + \epsilon)\text{dist}(x, y)$. Replacing T in F_0 with $R[x, y]$ produces a forest spanning the same set of terminals as F_0. Repeat this process for every monochromatic tree of F_0 to obtain a forest F_1 with Property 1.

Achieving the second property

Define a red terminal v to be a *red junction* of a forest F if there are two edges of R incident to v and both of them are spine edges of F. Define *blue junction* similarly.

Define a *witness path* for a forest U to be an R-to-B path P containing no edges of $R \cup B$ such that P neither starts nor ends at a junction.

Our goal in this section is to modify F_1 so as to bound the number of runs. We use the following lemma.

LEMMA 4.2. *For any forest F, the number of runs is at most the maximum number of vertex-disjoint witness paths.*

Proof. Let S be a spine of F. Arbitrarily orient S. For each run M belonging to S, we select a witness path P_M. The path P_M does not use the first vertex of M. It follows that the witness paths are vertex-disjoint.

Let M be a run, and let w be its last vertex. There are three cases.

Case I: Some internal vertex v of M is red-attached. Then w must be blue-attached. Define P_M as the concatenation of (1) a red-to-v path through a v-rooted off-spine red subtree, (2) the v-to-w subpath of M, and (3) a w-to-blue path through a w-rooted off-spine blue subtree.

Case II: Some internal vertex of M is blue-attached. This case is similar to Case I.

Case III: No internal vertex of M is either red-attached or blue-attached. In this case, w must be both red-attached and blue-attached. Define P_M as a red-to-w path through a w-rooted off-spine red subtree, concatenated with a w-to-blue path through a w-rooted off-spine blue subtree. $\square$

It remains to modify F_1 so as to bound the maximum number of vertex-disjoint witness paths.

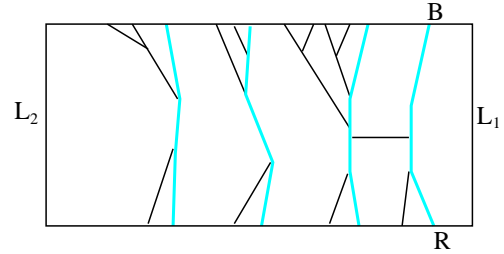Let $\mathcal{P}_1$ be a maximum set of vertex-disjoint witness paths in F_1.

Figure 4: A maximum set of vertex-disjoint witness paths, indicated as light lines.

Recall the vertices $r_0, r_1, r_2, \ldots, r_s$ discussed in the third of the panel properties. For each i, if there is a vertex x in $\{\text{start}(P) : P \in \mathcal{P}_1\}$ that lies in the interval $R[r_i, r_{i+1})$, let x_i be the rightmost such vertex. If there is no such vertex, let $x_i = r_i$.

We obtain F_2 from F_1 by adding the edges of $\bigcup_i R[r_i, x_i]$ and removing just enough edges not in R to make F_2 acyclic. Refer to Figure 5 for an illustration of the construction of F_2.

First we show that $\ell(F_2) \leq (1 + \epsilon)\ell(F_1)$. By a panel property, $\ell(R[r_i, x_i]) \leq \epsilon \cdot x_i$-to-blue distance. If $\ell(R[r_i, x_i]) > 0$ then there is a path in $\mathcal{P}_1$ whose length is at least the x_i-to-blue distance. This shows $\sum_i \ell(R[r_i, x_i]) \leq \epsilon\,\ell(F_1)$.

Let $\mathcal{P}_2$ be a maximum set of vertex-disjoint witness paths in F_2. We show that $|\mathcal{P}_2| \leq 6(k + 1)$ by showing that, for each i, there are at most two witness paths starting at vertices in $R[r_i, x_i]$ and at most four starting at vertices in $R(x_i, r_{i+1})$.

Fix i, and consider $R[r_i, x_i]$. Suppose F_2 has (at least) three witness paths Q_1, Q_2, Q_3 (ordered left to right by the positions on R of their start vertices) all starting in $R[r_i, x_i]$. Then $R[\text{start}(Q_1), \text{start}(Q_3)]$
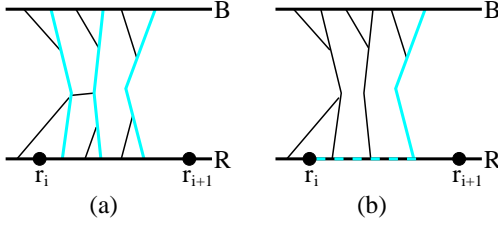
Figure 5: Construction of the forest F_2 that satisfies the second property. (a) A part of F_1 with witness paths (light) with red vertices between r_i and r_{i+1} on R. (b) The construction adds the light dashed path to F_2 and removes edges to ensure F_2 is a forest. There is only one witness path left.

consists of spine edges since each component of $F_2 - R[\text{start}(Q_1), \text{start}(Q_3)]$ contains both red and blue vertices (namely the endpoints of Q_1 and Q_3). Hence each vertex in $R(\text{start}(Q_1), \text{start}(Q_3))$ is a red junction, which contradicts the choice of the start of Q_2. Thus F_2 has at most two witness paths that start in $R[r_i, x_i]$.

Let $\mathcal{S}$ be the set of witness paths in $\mathcal{P}_2$ whose start vertices are in $R(x, r_{i+1})$. Assume for a contradiction that $|\mathcal{S}| \geq 5$. Let $\mathcal{A}$ be the set of paths in $\mathcal{P}_1$ that intersect paths in $\mathcal{S}$. Every witness path in F_2 is a witness path in F_1, so $\mathcal{P}_1 \cup \mathcal{S} - \mathcal{A}$ is a set of disjoint witness paths in F_1. Since $\mathcal{P}_1$ is maximum, $|\mathcal{A}| \geq 5$. By choice of x, none of the paths in $\mathcal{A}$ start in the interval $R(x, r_{i+1})$, so each starts either to the left of x or to the right of r_{i+1}. Hence either three or more start to the left of x or three or more start to the right of r_{i+1}. Assume the latter without loss of generality, and let A_1, A_2, A_3 be the three paths in right-to-left order of their start vertices on R.

Let $\mathcal{S}'$ be the subset of paths in $\mathcal{S}$ that intersect A_1, A_2, A_3. If $|\mathcal{S}'| < 3$ then $\mathcal{P}_2 \cup \{A_1, A_2, A_3\} - \mathcal{S}'$ would be a vertex-disjoint set of witness paths in F_2 of size greater than $|\mathcal{P}_2|$, a contradiction. Hence $|\mathcal{S}'| \geq 3$. Let S_1, S_2, S_3 be three paths in $\mathcal{S}'$, in right-to-left order of their start vertices on R.

Now S_1 and S_2 must intersect at least two of A_1, A_2, A_3 (using the fact that $|\mathcal{P}_1|$ is maximum). Because $\text{start}(S_2)$ is right of $\text{start}(S_1)$, S_2 must intersect A_2 and hence also A_1 (because $\text{start}(A_1)$ is between $\text{start}(S_2)$ and $\text{start}(A_2)$). Since $\text{start}(S_3)$ is right of $\text{start}(S_2)$, S_3 must intersect A_1 and A_2. Now there is a cycle in F_1 consisting of subpaths of $S_2, S_3, A_1,$ and A_2 between the vertices $V(A_2) \cap V(S_2)$, $V(A_2) \cap V(S)3$, $V(A_1) \cap V(S_2)$, and $V(A_1) \cap V(S_3)$. Since F_1 is a forest, this is a contradiction.

This gives us a total of $6(k+1) < c_1 \epsilon^{-3}$ witness paths in F_2 for a small constant c_1 and so F_2 has

Property 2.

Achieving the third property

F_2 has at most $c_1 \epsilon^{-3}$ runs. We modify F_2 obtaining F_3 so that for each of F_2's runs there are at most $\epsilon^{-1} + 6$ off-spine subtrees in F_2. The modifications preserve Property 1.

Consider in turn each run S of a spine of F_2. The start and end vertices of S are each roots of at most one red off-spine tree and one blue off-spine tree, for a total of four off-spine trees. Now consider the internal vertices of S. Assume without loss of generality that no internal vertex of S is blue-attached. Suppose that S contains at least $\epsilon^{-1} + 2$ red-attached vertices. In this case, we will perform a modification of the sort depicted in Figure 6. The operation will reroute the spine and eliminate all but at most $\epsilon^{-1} + 2$ of the off-spine subtrees. Let α_S be the sum of (a) the length of the subpath of S consisting of its internal vertices, and (b) the total length of the off-spine subtrees rooted at these vertices. We will show that, for a constant c, the increase in length is bounded by $c\epsilon \cdot \alpha_S$. (Thus the overall length increase due to all such modifications is at most $c\epsilon \, \ell(F_2)$.)
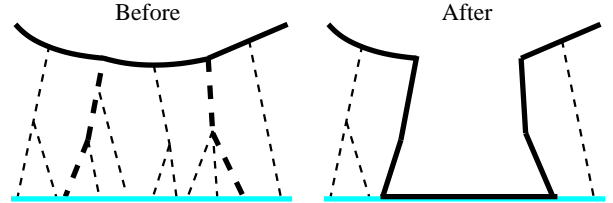

Figure 6: The spine (bold path) is redirected along theo leftmost path of an off-spine subtree (bold dashed path), along R (light path) and along the reverse of the rightmost path of another off-spine subtree (bold dashed path) allowing us to remove the off-spine trees between these two.

Let $T_1, \ldots, T_s$ be the off-spine red subtrees rooted at vertices of S, ordered left to right by the positions on R of their leaves. Let R' be the minimal subpath of R containing all these leaves. Consider the path from the leftmost vertex of R' that goes along T_1 to its root, then along the spine, then down T_s to the rightmost vertex of R'. By a panel property, the length of this path times $1 + \epsilon$ is at least $\ell(R')$, so $(1+\epsilon)\alpha_S \geq \ell(R)$ (This fact is used in Case II below.)

Say a subtree T_i is *light* if its weight is less than $\epsilon \, \ell(R')$ and is *heavy* otherwise. Let $L = \{i : T_i \text{ is light}\}$. Let

$$\tau = \begin{cases} (\min L) - 1 + s - (\max L) & \text{if } L \neq \emptyset \\ s & \text{otherwise} \end{cases}$$
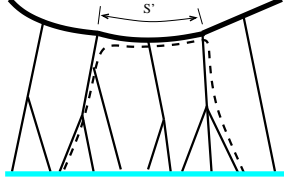
Figure 7: The path P is obtained by traversing the leftmost path of $T_{\min L}$ to the root, then along the spine to the root of $T_{\max L}$, then down the rightmost path. The subpath of spine traversed is S'. Because of a panel property, the subpath of R between the start and end of P is almost as short as P.

Case I: $\tau \geq \epsilon^{-1} + 2$. In this case, the operation depicted in Figure 6 is applied to T_1 and T_s. This operation retains the leftmost path of T_1 and the rightmost path of T_s, and adds the subpath R', and these paths now become part of the spine. The operation removes all other edges of T_1 and T_s, and all edges in $T_2, \ldots, T_{s-1}$. The connectivity between the leaves of the T_i's and the spine is preserved because of the addition of R'. The resulting decrease in weight is at least

$$
\begin{aligned}
(\sum_{i=2}^{s-1} \ell(T_i)) - \ell(R') &\geq (\epsilon^{-1} \, \epsilon \ell(R')) - \ell(R') \\
&\geq 0
\end{aligned}
$$

Case II: $\tau < \epsilon^{-1} + 2$ In this case, the operation depicted in Figure 6 is applied to $T_{\min L}$ and $T_{\max L}$. This operation retains the leftmost path P_1 of $T_{\min L}$ and the rightmost path P_2 of $T_{\max L}$, and adds the subpath $R[\text{end}(P_1), \text{end}(P_2)]$. The operation removes all other edges of $T_{\min L}$ and $T_{\max L}$, and removes all edges in $T_{1+\min L}, T_{2+\min L}, \ldots, T_{-1+\max L}$. The operation also removes the subpath S' of the spine from the root of $T_{\min L}$ to the root of $T_{\max L}$. The operation leaves at most $\epsilon^{-1} + 2$ off-spine subtrees.

Consider the path P obtained (as shown in Figure 7 by traversing P_1 in reverse (towards S), then traversing the subpath S', then traversing P_2 back to R. Note that $\ell(P) = \ell(P_1) + \ell(S') + \ell(P_2)$. By a panel property, $\ell(R[\text{end}(P_1), \text{end}(P_2)]) \leq (1 + \epsilon)\ell(P)$. The increase in length due to this operation is

$$
\begin{aligned}
\ell&(R[\text{end}(P_1), \text{end}(P_2)]) - \ell(S') \\
&\leq (1 + \epsilon)(\ell(P_1) + \ell(S') + \ell(P_2)) - \ell(S') \\
&\leq 2\epsilon(1 + \epsilon)\ell(R') + \epsilon \ell(S')
\end{aligned}
$$

Clearly the length of the subpath of S consisting of its internal vertices is at least $\ell(S')$. This shows $\alpha_S \geq \ell(S')$. By the fact mentioned before the case analysis

began, $(1 + \epsilon)\alpha_S \geq \ell(R')$. Combining these inequalities shows that the increase in length due to the operation is at most $c\epsilon \, \alpha_S$ for some constant c.

Achieving the fourth property

Let T be a off-spine subtree of F_3 that is rooted at a spine vertex and whose leaves are terminals on (without loss of generality) R.

As long as there exists a vertex with at least three children, carry out the following step. Let u be a vertex of T with at least three children such that u has no such ancestor in T. let T_u be the subtree of T rooted at u and replace T_u with (1) the subpath R' of R that spans all terminals of T_u and (2) the shortest u-to-R' path (Figure 8 (a)).

Let T' be the tree thereby obtained. Every vertex has at most two children in T'.

Partition the edges of T' into *super-edges*, defined as follows: a super-edge is a maximal path in T' whose internal vertices all have degree 2 in T'. Partition the super-edges into levels according to the number of super-edges traversed when going from the root to the beginning of the super-edge. There exists a level $i \in \{1, \ldots, \epsilon^{-1}\}$ such that the sum of the costs of the super-edges at level i is at most $\epsilon \ell(T')$.

Let $\mathcal{U}$ be the set of all vertices which are start vertices of super-edges of T'' at level i. For each $u \in \mathcal{U}$, let T'_u be the subtree of T' rooted at u, and replace T'_u with (1) the subpath R' of R spanning all terminals of T'_u and (2) the shortest u-to-R' path (Figure 8 (b)).
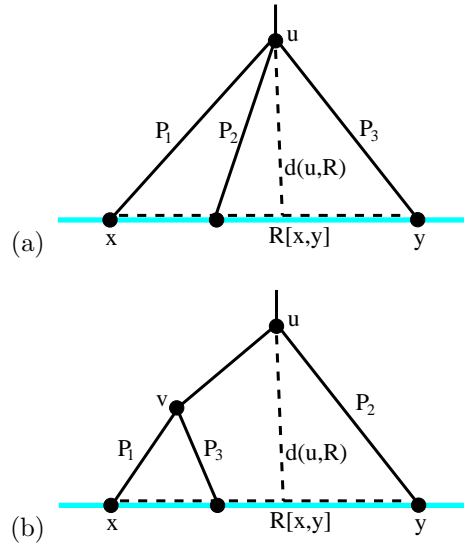

Figure 8: (a) and (b) Replacing the tree rooted at u with $R[x, y]$ and a path of length $\text{dist}(u, R)$ is cheap.

Let T'' be the tree thereby obtained. Since each vertex of T'' has at most two children, and T'' has at most $1/\epsilon$ levels, T'' has at most $2^{1/\epsilon+1}$ vertices of degree 1 on R. Thus T'' satisfies Property 4.

Now we analyze the increase in cost.

First, we analyze the increase in cost when going from T to T'. Consider a single replacement step. Note that T_u has cost equal to the distance from u to R' plus the length of R'. Since u has at least three children in T, it must be that T_u contains three disjoint paths from u to R: a path P_L to the leftmost terminal x reached, a path P_R to the rightmost terminal y reached, and a path P_M to some terminal in the middle (Figure 8(a)). Note that $\ell(P_L) + \ell(P_R)$ must be at least the distance between x and y, which is at least $\ell(R')/(1+\epsilon)$ by one of the panel properties. Moreover, $\ell(P_M) \geq \text{dist}(u, R')$. It follows that the length of the set T_u of edges removed is at least $\ell(R')/(1+\epsilon) + \text{dist}(u, R')$, which in turn is least $1/(1+\epsilon)$ times the length of the replacement paths. Summing over all replacements, we infer that $\ell(T') \leq (1+\epsilon)\ell(T)$.

Second, we analyze the increase in cost when going from T' to T''. Let p be a super-edge from u to v. Since u has at least three children, it must be that T'_u contains three disjoint paths: a path P_L from u to the leftmost terminal x reached, a path P_R from u to the rightmost terminal y reached, and a path P_M from v to some terminal in the middle (Figure 8(b)). We have: $\ell(P_L) + \ell(P_R) \geq \text{dist}(x, y) \geq \ell(R')/(1+\epsilon)$. Moreover, $\ell(P_M) \geq \text{dist}(v, R') \geq \text{dist}(u, R') - \ell(p)$. Thus:

$$\begin{aligned} \ell(T'_u) &\geq \ell(R')/(1+\epsilon) + \text{dist}(u, R') - \ell(p) \\ &\geq \ell(T'_u)/(1+\epsilon) - \ell(p). \end{aligned}$$

Summing, we get that $\ell(T'') \leq \ell(T')(1+\epsilon) + \sum \ell(p) \leq \ell(T')(1+\epsilon) + \epsilon\ell(T)$, so $\ell(T'') \leq (1+4\epsilon)\ell(T)$.

Repeating this process for every off-spine subtree of F_3 gives a forest F_4 satisfying Property 4. It is clear that the process does not increase the number of off-spine trees or create monochromatic trees.

References

[1] S. Arora. Polynomial-time approximation schemes for euclidean TSP and other geometric problems. *JACM*, 45(5):753–782, 1998.

[2] S. Arora, M. Grigni, D. Karger, P. Klein, and A. Woloszyn. A polynomial-time approximation scheme for weighted planar graph TSP. In *9th SODA*, pages 33–41, 1998.

[3] B. Baker. Approximation algorithms for NP-complete problems on planar graphs. *JACM*, 41(1):153–180, 1994.

[4] M. Bern. Faster exact algorithms for steiner trees in planar networks. *Networks*, 20:109–120, 1990.

[5] M. Bern and D. Bienstock. Polynomially solvable special cases of the steiner problem in planar networks. *Annals of Operations Research*, 33:405–418, 1991.

[6] M. Bern and P. Plassmann. The Steiner problem with edge lengths 1 and 2. *IPL*, 32:171–176, 1989.

[7] R. E. Erickson, C. L. Monma, and A. F. Veinott. Send-and-split method for minimum concave-cost network flows. *Mathematics of Operations Research*, 12:634–664, 1987.

[8] I. Hicks, A. Koster, and E. Kolotoğlu. Branch and tree decomposition techniques for discrete optimization. In J. Smith, editor, *TutORials 2005*, INFORMS TutO-Rials in Operations Research Series, chapter 1, pages 1–29. INFORMS Annual Meeting, 2005.

[9] P. Klein. A linear-time approximation scheme for planar weighted TSP. In *46th FOCS*, pages 647–647, 2005.

[10] P. Klein. A subset spanner for planar graphs, with application to subset TSP. In *38th STOC*, pages 749–756, 2006.

[11] L. Kou, G. Markowsky, , and L. Berman. A fast algorithm for Steiner trees. *Acta Informatica*, 15:141–145, 1981.

[12] K. Mehlhorn. Approximation algorithm for the Steiner problem in graphs. *IPL*, 27(3):125–128, 1988.

[13] J. S. Provan. An approximation scheme for finding steiner trees with obstacles. *SIAM Journal on Computing*, 17:920–934, 1988.

[14] J. S. Provan. Convexity and the steiner tree problem. *Networks*, 1988.

[15] S. Rao and W. Smith. Approximating geometrical graphs via "spanners" and "banyans". In *ACM Symposium on Theory of Computing*, pages 540–550, 1998.

[16] H. Takahashi and A. Matsuyama. An approximate solution for the steiner problem in graphs. *Mathematica Japonicae*, 24:571–577, 1980.

[17] 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.

[18] Y. Wu, P. Widmayer, and C. Wong. A faster approximation algorithm for the Steiner problem in graphs. *Acta informatica*, 23(2):223–229, 1986.