Node-Weighted Steiner Tree and Group Steiner Tree in Planar Graphs

Erik D. Demaine[1], MohammadTaghi Hajiaghayi[2], and Philip N. Klein[3]

[1] MIT Computer Science and Artificial Intelligence Laboratory,
32 Vassar St., Cambridge, MA 02139, USA, edemaine@mit.edu
[2] AT&T Labs — Research,
180 Park Ave., Florham Park, NJ 07932, USA, hajiagha@research.att.com
[3] Department of Computer Science, Brown University,
Providence, RI 02912, USA, klein@cs.brown.edu

Abstract. We improve the approximation ratios for two optimization problems in planar graphs. For node-weighted Steiner tree, a classical network-optimization problem, the best achievable approximation ratio in general graphs is $\Theta(\log n)$, and nothing better was previously known for planar graphs. We give a constant-factor approximation for planar graphs. Our algorithm generalizes to allow as input any nontrivial minor-closed graph family, and also generalizes to address other optimization problems such as Steiner forest, prize-collecting Steiner tree, and network-formation games.

The second problem we address is group Steiner tree: given a graph with edge weights and a collection of groups (subsets of nodes), find a minimum-weight connected subgraph that includes at least one node from each group. The best approximation ratio known in general graphs is $O(\log^3 n)$, or $O(\log^2 n)$ when the host graph is a tree. We obtain an $O(\log n \,\mathrm{polyloglog}\, n)$ approximation algorithm for the special case where the graph is planar embedded and each group is the set of nodes on a face. We obtain the same approximation ratio for the minimum-weight tour that must visit each group.

1 Introduction

One of the most fundamental problems in combinatorial optimization is the *network Steiner tree problem*. This was one of the first problems shown NP-hard by Karp [19]. In the traditional formulation, we are given an undirected graph with edge costs and a subset of nodes called *terminals*. The goal is to find a minimum-cost subgraph of G that connects the terminals. A long sequence of papers give polynomial-time constant-factor approximation algorithms for this problem; the current best approximation ratio is 1.55 [3].

The generalization[4] of network Steiner tree in which the nodes are also assigned costs is of both practical and theoretical significance. On the practical side, in telecommunications for example, expensive equipment such as routers

[4] The case of both edge costs and node costs can be reduced to the case of node costs.

and switches are at the nodes of the underlying network and it is natural to model such problems as node-weighted. On the theoretical side, node-weighted versions of many classic edge-weighted problems have been considered by many authors so far; see, e.g., [8,10,15,17,22] for some recent work.

Unfortunately, set cover can be reduced to node-weighted Steiner tree in general graphs, so an approximation ratio better than $\ln n$ is not achievable in polynomial time unless P = NP [9,26]. Klein and Ravi [20] gave a polynomial-time approximation algorithm with a performance ratio of $O(\log n)$, so this result is within a constant factor of optimal. To achieve a better bound, we must restrict the class of inputs.

A natural restriction is planarity. In practical scenarios of physical networking, with cable or fiber embedded in the ground, crossings are rare or nonexistent. Somewhat surprisingly, no one has yet addressed this classic problem of node-weighted Steiner tree in this natural class of graphs. In this paper, we achieve a much better approximation ratio for this problem:

Theorem 1. *There is a polynomial-time* 6*-approximation algorithm for node-weighted Steiner tree in planar graphs.*

Our algorithm is a simple and natural extension of primal-dual techniques to node weights, which to our knowledge has never been explored. We suspect that the factor 6 can be improved.

In fact, our result is more general in two senses: we show that a constant approximation ratio is achievable for a much broader family of graphs, and we show that a much more general optimization problem can be approximated.

1.1 Broader Graph Classes. A *minor* of a graph G is a graph obtainable from G by deleting and contracting edges.

It is well-known that planar graphs are the graphs with no minor isomorphic to $K_{3,3}$ or K_5. More generally, for any graph H, we can consider the family of graphs excluding H as a minor. We obtain the following generalization of Theorem 1.

Theorem 2. *For any graph H, there is a constant c_H and a polynomial-time c_H-approximation algorithm for node-weighted Steiner tree on H-minor-free graphs.*

1.2 More Network-Design Problems. Our algorithm can solve a broader range of network-design problems. The node-weighted *Steiner forest problem* takes as input an undirected graph with node costs and a set of unordered pairs $\{s_i, t_i\}$ of nodes. The goal is to find the minimum-cost network that includes a path between each given pair of nodes. For the edge-weighted case, there is a polynomial-time 2-approximation algorithm [2]. For the node-weighted case, an $O(\log n)$-approximation can be achieved [20]. In this paper, we achieve a constant-factor approximation for planar graphs, and more generally, graphs excluding a fixed minor.

Theorem 3. *There is a polynomial-time* 6*-approximation algorithm for node-weighted Steiner forest in planar graphs. For any graph H, there is a constant c_H and a polynomial-time c_H-approximation algorithm for this problem on H-minor-free graphs.*

More generally, in Section 2, we show how our framework applies to a node-weighted variation of *proper 0-1 functions* by Goemans and Williamson [12]. These functions model node-weighted versions of several other problems, e.g., T-joins and nonfixed point-to-point connections. We thus obtain constant-factor approximation algorithms for all of these problems when the input is restricted to be a planar graph or a graph excluding a fixed minor.

We also consider the *prize-collecting* version of Steiner tree, where some terminal pairs can remain disconnected, but we pay a specific penalty for each such pair. The best approximation algorithm for this problem achieves an $O(\log n)$ approximation ratio [25]. (See also the related work on the dual quota version of the problem [15,25].) Similar to [7,16], we can prove the following:

Theorem 4. *The prize-collecting Steiner forest problem has a constant-factor approximation algorithm in graphs excluding a fixed minor.*

1.3 Other Applications. Node-weighted Steiner tree is a network design problem with many practical applications and theoretical implications. Enumerating such applications is beyond the scope of this paper. We point out, however, that it has an application even in network formation games.

Anshelevich et al. [4] consider a network formation game in which k terminals (players) buy edges in a directed graph, equally sharing the unit cost of an edge bought by multiple players, to form a Steiner tree. They prove that the price of stability in this game is at most H_k (the kth harmonic number, which is within an additive 1 of $\ln k$) by defining a dynamics that converges to an equilibrium within an H_k factor of the social optimum. However, their dynamics [4, Theorem 2.2] starts by computing an optimal Steiner tree, which cannot even be efficiently approximated. With our results, we can obtain a polynomially computable Nash equilibrium within an H_k factor of the social optimum for their game in node-weighted undirected planar graphs. Furthermore, this bound is tight: there is a node-weighted graph whose only Nash equilibrium is a factor H_k worse than the social optimum.[5]

1.4 Planar Group Steiner Tree. In the wire-routing phase of VLSI design, a *net* is a set of pins on the boundaries of various components that must be connected. A minimum-length Steiner tree is a natural choice for routing the net. (The routing must avoid previously routed nets and other obstacles.) Reich and Widmayer [27] observed that, for each component, there is flexibility as to

[5] The graph has terminals $t_1, t_2, \ldots, t_k$, additional nodes $u_1, u_2, \ldots, u_k$, where u_i has weight $1/i$, and another node v of weight $1 + \varepsilon$. Each terminal t_i has two candidate paths to the root r, one through u_i and the other through v. (This construction is similar to [4, Figure 1].) The social optimum buys v at cost $1 + \varepsilon$, but the Nash equilibrium buys $u_1, u_2, \ldots, u_k$ at cost H_k.

the location of the pin used, and that the routing of the net should exploit that flexibility.

With that as motivation, they introduced the *group Steiner tree problem*: we are given a graph G with edge weights, and a collection $g_1, g_2, \ldots, g_k$ of node sets called *groups*. The goal is to find a minimum-weight connected subgraph of G that contains at least one node from each group.

Much research has gone into finding good approximation algorithms for this problem. For general graphs, the best approximation ratio known to be achievable in polynomial time [11] is $O(\log^3 n)$, and for trees, the best known is $O(\log^2 n)$.

Even when the host graph is a tree and hence planar, the problem cannot be approximated better than $\Omega(\log^{2-\varepsilon} n)$ unless NP admits quasipolynomial-time Las Vegas algorithms [18]. It would thus appear that restricting the input to planar graphs cannot lead to a substantially improved approximation.

Returning to the origin of the problem provides some inspiration. In a VLSI instance, the elements of a single group are all located on the boundary of a component, which occupies a region on the plane. Motivated by this real-world restriction, we define an instance of the *planar group Steiner tree problem* to be a planar embedded graph G with edge weights, and a collection of groups $g_1, g_2, \ldots, g_k$ and corresponding distinct faces $f_1, f_2, \ldots, f_k$ of G, such that the nodes belonging to each group g_i lie on the boundary of the corresponding face f_i.

We can without loss of generality require that each group g_i consists of exactly the nodes on the boundary of f_i. (The more general problem can be reduced to this one by the introduction of high-weight edges.) Therefore, an equivalent and more concise definition of an instance of *planar group Steiner tree* is a planar embedded graph G and a set of faces $f_1, f_2, \ldots, f_k$, which implicitly define a group for each f_i consisting of the nodes on the boundary of f_i.

Theorem 5. *Planar group Steiner tree has a polynomial-time $O(\log n \operatorname{polyloglog} n)$-approximation algorithm.*

Our proof of this theorem uses probabilistic embedding into spanning trees with expected distortion $O(\log n \log\log n (\log\log\log n)^3)$ [1]. (We cannot use the original result of Bartal [6] which does not preserve the planar structure of the problem.) On trees, we can solve the problem via dynamic programming. Alternatively, because paths in trees cannot properly cross, we can use the following rounding result of independent interest:

Theorem 6. *Any solution f to the noncrossing-flow relaxation of directed Steiner tree can be converted in polynomial time into an integral solution f'' of weight $c(f'') \leq 6\, c(f)$.*

Recall the *directed Steiner tree problem*: we are given a directed graph G with edge costs, a sink node s, and a set T of *terminal* nodes. The goal is to find a minimum-cost subgraph of G that, for each terminal t_i, contains a directed path from t_i to s. The noncrossing-flow variation is a natural relaxation of directed Steiner tree in planar graphs, a kind of minimum-cost flow where flow paths cannot cross; see Appendix A. Using a novel approach for rounding such *planar* flows, we show that a (fractional) solution to this noncrossing relaxation can be converted into an (integral) directed Steiner tree whose cost is at most 6 times the value of the solution to the relaxation.

Unfortunately, we do not know a polynomial-time algorithm for finding an optimal solution to the noncrossing-flow formulation for an arbitrary planar instance of directed Steiner tree. Such a result would yield a constant-factor approximation algorithm for planar directed Steiner tree.

Related Work. Motivated in part by the VLSI application, Mata and Mitchell [23] consider the following problem: given a set of n polygonal regions in the plane, find a tour that visits at least one point from each region. They describe this problem as a special case of the problem *TSP with neighborhoods* (also called *group TSP*). They give a polynomial-time $O(\log n)$-approximation algorithm. Because the tour contains a spanning tree, and doubling each edge of a tree yields a tour, it is also an approximation algorithm for group Steiner tree where the groups are the polygonal regions. Gudmundsson and Levcopoulos [14] gave a faster algorithm for the same problem. No known polynomial-time algorithm achieves an approximation ratio better than $\Theta(\log n)$ for this problem. On the lower-bound side, unless P = NP, no constant-factor approximation is possible for disjoint disconnected regions, and no $(2-\varepsilon)$-approximation is possible for (nondisjoint) connected regions [28].

Arkin and Hassin [5] gave constant-factor approximation algorithms for the special cases of parallel unit-length line segments, translated copies of a polygonal region, and circles. Mitchell [24] recently gave a PTAS for group TSP when the groups are disjoint and "fat".

Our bound for planar group Steiner tree nearly matches the bound of Mata and Mitchell. Our approach has the advantage that planar graphs can capture metrics that are not captured by the Euclidean metric, useful e.g. in the VLSI problem where the routing of a net must avoid obstacles and previously routed nets.

2 Node-Weighted Network Design in Planar and Minor-Excluding Graphs

For a graph G and a set S of nodes, we use $G[S]$ to denote the subgraph of G induced by the nodes of S.

To formulate the node-weighted network-design problems we address, we adapt an approach due to Goemans and Williamson [12].

Proper Function. Let V be the set of nodes in an undirected graph G. A function $f : 2^V \to \{0,1\}$ is *proper* if $f(\emptyset) = 0$ and the following two properties hold:

1. (Symmetry) $f(S) = f(V - S)$.
2. (Disjointness) If S_1 and S_2 are disjoint, then $f(S_1) = f(S_2) = 0$ implies $f(S_1 \cup S_2) = 0$.

In [12], proper functions are used in a formulation of *edge-weighted* network-design problems as *cut-covering* problems. A proper function specifies a family of cuts, and the goal is to minimize the cost of a set of edges that covers all cuts in this family.

Following Klein and Ravi [20], we adapt this formulation for *node-weighted* problems. We use $\Gamma(S)$ to denote the set of nodes that are not in S but have neighbors in S. The problems we address can be formulated by the following integer linear program with a variable $x(v)$ for each node $v \in V$:

$$\begin{aligned} &\text{minimize } \textstyle\sum_{v\in V} w(v)\, x(v) \\ &\text{subject to } \textstyle\sum_{v\in \Gamma(S)} x(v) \geq f(S) \quad \text{for all } S \subseteq V, \\ &\qquad\qquad x(v) \in \{0,1\} \qquad\quad \text{for all } v \in V. \end{aligned} \tag{1}$$

where f is a proper 0-1 function. The minimum solution x to this integer program assigns 1's to a subset X of nodes, and X is then considered the solution to the network-design problem. Conversely, a subset X of nodes is considered a *feasible solution* if the corresponding $\{0,1\}$-assignment to nodes of G satisfies the inequalities.

For example, consider the node-weighted Steiner forest problem. The input is an undirected graph G with node weights $w(v)$, and a set of pairs $\{s_i, t_i\}$ of nodes. For a set S of nodes, define $f(S)$ to be 1 if, for some pair $\{s_i, t_i\}$, S contains one element of the pair but not the other. Otherwise, define $f(S)$ to be 0. It is easy to verify that this function is proper. To see that the solution to the integer linear program is a solution to the Steiner forest instance, assume for a contradiction that some pair s_i, t_i are not connected via nodes assigned 1's by x. Let S be the set of nodes connected to s_i via such nodes. By our assumption, $f(S) \geq 1$ but by the choice of S, every node $v \in \Gamma(S)$ is assigned 0 by x, contradicting the linear constraint.

We assume in our algorithm that $f(\cdot)$ can be queried for a specific set S in polynomial time. For the analysis, we assume that each node v with $f(\{v\}) = 1$ has zero cost. For the Steiner forest problem, for example, each such node belongs to some pair, so must belong to the solution, so we can make this assumption without loss of generality.

For a subset X of nodes in a graph G, let $\mathrm{CC}(X)$ denote the node sets of connected components of the subgraph of G induced by X.

Lemma 1. *Let X be a subset of nodes of G that contains every node v such that $f(\{v\}) = 1$. Suppose that $f(C) = 0$ for every $C \in \mathrm{CC}(X)$. Then X is a feasible solution to the integer program.*

Proof. Let x be the function that assigns 1 to nodes in X and 0 to nodes not in X. Let S be any subset of nodes such that $\sum_{v\in\Gamma(S)} x(v) = 0$. We shall show that $f(S) = 0$.

For each $C \in \mathrm{CC}(X)$, we claim that $S \cap C$ is either $\emptyset$ or C. (Otherwise, there would be a pair u, v of adjacent nodes in X such that $u \in S$ and $v \notin S$, so $v \in \Gamma(S)$. Since $x(v) = 1$, this would contradict the choice of S.)

The claim implies that S is the disjoint union of some of the connected components $C \in \mathrm{CC}(X)$ together with some singletons $\{v\}$ with $v \notin X$. We assumed that X contains all nodes v with $f(\{v\}) = 1$, so $f(\{v\}) = 0$ for $v \notin X$. We also assumed that each connected component C of X has $f(C) = 0$. Combining these facts using the disjointness property of f, we infer that $f(S) = 0$. □

Dual. The linear relaxation of the above integer program is obtained by replacing the constraint $x(v) \in \{0, 1\}$ with the constraint $x(v) \geq 0$. The dual of the resulting linear program is as follows:

$$\begin{array}{rll} \text{maximize} & \sum_{S \subseteq V} f(S)\, y(S) & \\ \text{subject to} & \sum_{S \subseteq V : v \in \Gamma(S)} y(S) \leq w(v) & \text{for all } v \in V, \\ & y(S) \geq 0 & \text{for all } S \subseteq V. \end{array}$$

There is a dual variable $y(S)$ for each subset S of V. However, the only such variables that affect the objective function (and therefore the only variables we need to consider) are those variables $y(S)$ where $f(S) = 1$. Intuitively, the goal of the dual linear program is to find a maximum-size family of node sets S with $f(S) = 1$ subject to the constraint that each node v is the neighbor of at most $w(v)$ sets in the family.

Primal-Dual Algorithm. Goemans and Williamson [13] gave a generic version of the primal-dual approximation algorithm. In this section, we give an algorithm that is a specialization (and slight modification) of their generic algorithm. We start with some terminology.

Let G be the input graph. A node set S is a *violated connected component with respect to* X if $S \in \mathrm{CC}(X)$ and $f(S) = 1$. Define a *partial solution* to be a set X of nodes containing $\{v : f(v) = 1\}$ such that there is some violated connected component with respect to X.

Goemans and Williamson's generic algorithm is defined in terms of an oracle. We will use an oracle $\mathrm{Viol}(\cdot)$ that takes a partial solution X as input, and that outputs the violated connected components with respect to X. This oracle can be implemented in polynomial time using a connected-components subroutine and queries to the function $f(\cdot)$.

Now we give our specialization and modification of the generic algorithm. The modification is as follows. Their algorithm maintains a solution X, initially empty, and adds to it in a series of iterations; finally, the algorithm removes some elements from it. In our modified version, X initially consists of all nodes v such that $f(\{v\}) = 1$, and these elements are never removed from X. However, these nodes are required to have weight zero, so their presence in X does not affect the approximation performance.

The algorithm also maintains a dual feasible solution $\mathbf{y}$. Recall that the dual linear program has a constraint $\sum_{S \subseteq V : v \in \Gamma(S)} y(S) \leq w(v)$ for each node $v \in V$.

1. $\mathbf{y} \leftarrow \mathbf{0}$
2. $X \leftarrow \{v \ : \ f(\{v\}) = 1\}$
3. While there is a violated connected component with respect to X:
   (a) Increase $y(S)$ uniformly for all sets $S \in \mathrm{Viol}(X)$ until the dual linear-program inequality for some v becomes tight: $\sum_{S \subseteq V : v \in \Gamma(S)} y(S) = w(v)$.
   (b) Add v to X, i.e., $x(v) \leftarrow 1$.
4. For each v in X in the reverse of the order in which they were added during the while-loop:
   (a) If $f(S) = 0$ for every connected component $S \in \mathrm{CC}(X - \{v\})$, then remove v from X.
5. Return X.

The algorithm above is almost an instantiation of an algorithm of Goemans and Williamson [13]. The difference is that, in our algorithm, X is required at all times to contain every node v such that $f(\{v\}) = 1$. Because these nodes are assumed to have zero cost, the proof of Theorem 4.2 of [13] can be adapted to show the theorem below (see the full paper). For a set X of nodes of G, a set F of nodes is a *feasible augmentation of* X if $F \supseteq X$ and F is a feasible solution. If in addition no proper subset of F is a feasible augmentation of X then F is a *minimal* feasible augmentation of X.

Theorem 7. *Suppose γ is a number such that, for any partial solution $X \subseteq V$ and any minimal feasible augmentation F of X, we have*

$$\sum \{|F \cap \Gamma(S)| \ : \ S \in \mathrm{Viol}(X)\} \leq \gamma \, |\mathrm{Viol}(X)|. \tag{2}$$

Then the algorithm described above returns a feasible solution of weight at most $\gamma \sum_{S \subseteq V} y(S) \leq \gamma \, LP\text{-}OPT$ where $LP\text{-}OPT$ denotes the weight of an optimal solution to the linear program (1).

In order to apply Theorem 7, we need to prove (2) for some γ.

Theorem 8. *Let X be a partial solution, and let F be any minimal feasible augmentation of X.*

If G is planar, then $\sum \{|F \cap \Gamma(S)| \ : \ S \in \mathrm{Viol}(X)\} \leq 6 \, |\mathrm{Viol}(X)|$.

If G is H-minor-free, then $\sum \{|F \cap \Gamma(S)| \ : \ S \in \mathrm{Viol}(X)\} \leq O(|V(H)| \sqrt{\log |V(H)|}) \, |\mathrm{Viol}(X)|$.

The proof of Theorem 8 will be given in this section. By using the bounds proved in Theorem 8 in Theorem 7, we obtain

Theorem 9. *The primal-dual algorithm above is a 6-approximation on planar graphs and, more generally, an $O(1)$-approximation in H-minor-free graphs for any fixed graph H.*

Now we give the proof of Theorem 8.

The sum $\sum \{|F \cap \Gamma(S)| \ : \ S \in \mathrm{Viol}(X)\}$ counts the number of adjacencies between F and violated connected components of X, counting multiply if one violated connected component of X is adjacent to several nodes in F, but counting only once if multiple nodes in a common violated connected component of X are adjacent to one node of F.

Let $\hat{F} = F - X$. For any $S \in \mathrm{Viol}(X)$, since S is the node set of a connected component of $G[X]$, no neighbor of S belongs to X. This shows that $|\hat{F} \cap \Gamma(S)| = |F \cap \Gamma(S)|$. To prove Theorem 8, it therefore suffices to bound

$$\sum\{|\hat{F} \cap \Gamma(S)| \ : \ S \in \mathrm{Viol}(X)\}. \tag{3}$$

Let $\hat{G}$ be the graph obtained from G by contracting each violated connected component of X to a single node, which we call a *terminal*. Let R be the set of terminals. Because of the contractions, no two nodes of R are adjacent in $\hat{G}$. We discard multiple copies of edges in $\hat{G}$ so that the sum (3) in G becomes the number of edges in $\hat{G}$ between nodes in $\hat{F}$ and terminals. Our goal is to bound the number of such edges in terms of $|R|$. We do this separately for each connected component of $\hat{G}[\hat{F} \cup R]$. Let $G' = \hat{G}[F' \cup R']$ be one such connected component, where $F' \subseteq \hat{F}$ and $R' \subseteq R$.

By minimality of F, R' is nonempty. Assign distinct integers as IDs to the nodes of G'. Let r be a node in R'. For each node v in $F' \cup R'$, define its *level* $\ell(v)$ to be its breadth-first-search distance from r in G'. We next define the *parent* $p(v)$ of each node $v \neq r$. For $v \in R' - \{r\}$, define $p(v)$ to be a neighbor of v in G' having level $\ell(v) - 1$, namely that neighbor having minimum ID. For each node $v \in F'$, select $p(v)$ as follows. If v has a neighbor w in R' such that $p(w) \neq v$, then $p(v)$ is any such neighbor. Suppose that v has no neighbor w in R' such that $p(w) \neq v$. By the properties of breadth-first-search distances, v has some neighbor w' such that $\ell(w') = \ell(v) - 1$. Let $p(v)$ be this node w'. We show that in this case that $w' \in F'$.

Assume for a contradiction that $w' \in R'$. Its parent would have level $\ell(w') - 1 = \ell(v) - 2$, so its parent could not be v. This implies that v has a neighbor in R' whose parent is not v, a contradiction.

Lemma 2. *The parent pointers do not form a cycle.*

Proof. Suppose for contradiction that $C = x_0 x_1 x_2 \ldots x_{k-1} x_0$ were a minimal (simple) cycle with $p(x_i) = x_{i+1}$ for all i (where the indices are taken modulo k). The root r cannot be in the cycle C because it has no parent. Because $x_i x_{i+1}$ is an edge of the graph, $\ell(x_{i+1}) \leq \ell(x_i) + 1$ for every i. By construction, $\ell(x_{i+1}) = \ell(x_i) - 1$ for each i where $x_i \in R'$ and for each i where $x_i \in F'$ and $x_{i+1} \in F'$, So the only case in which $\ell(x_{i+1})$ could be $\ell(x_i) + 1$ is when $x_i \in F'$ and $x_{i+1} \in R'$. But then by construction $\ell(x_{i+2}) = \ell(x_{i+1}) - 1 = \ell(x_i)$. Thus every increase in level is immediately followed in the cycle C by a strict decrease in level.

Suppose there were two consecutive strict decreases: $\ell(x_{i+1}) = \ell(x_i) - 1$ and $\ell(x_{i+2}) = \ell(x_{i+1}) - 1$. Then all nodes after x_{i+2} would have level at most $\ell(x_i) - 1$, contradicting the fact that C is a cycle. It follows that the x_i's must alternate between R' and F', and that, for some positive integer d, the levels of the nodes in $C \cap R'$ are d, and the levels of the nodes in $C \cap F'$ are $d - 1$. Now consider the minimum-ID node x_i in $C \cap F'$. By construction, x_{i-1} and x_{i+1} must have

x_i as their parent. But this contradicts $p(x_{i+1}) = x_{i+2}$ given that the cycle is simple. $\square$

Lemma 2 shows that the parent pointers define a rooted spanning tree of $\hat{G}[F' \cup R']$. Let F'' be the subset of nodes in F' with a neighbor in R'. We need to compute the number of edges in $\hat{G}[F'' \cup R']$.

Lemma 3. $|F''| \leq 2\,|R'|$.

Proof. As we show in Figure 1, to each node u of R', we charge at most two nodes of F'': u's parent and the nearest ancestor of u whose parent is in R'. We claim that every node of F'' gets counted by this charging.

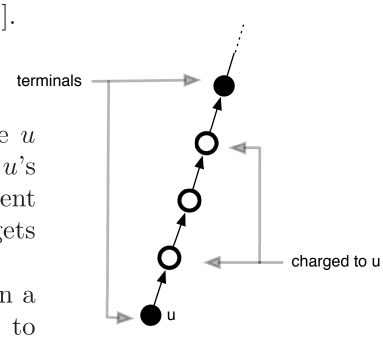

Fig. 1. Charging nodes of F'' to terminals.

By minimality of F, every node of F' is on a path in the tree T from some terminal in R' to the root r. Let v be any node of F''. Let P be the shortest terminal-to-terminal path in T that includes v. Since $v \in F''$, v has a neighbor w in R. If v is the second node of P then v is charged to the first node of P. Otherwise, by minimality of P, v is not the parent of a terminal, so $p(w) \neq v$, so v's parent is a terminal (not necessarily w), so again v is charged to the first node of P. $\square$

Now we can complete the proof of Theorem 8. Since G is H-minor free, and $\hat{G}[F'' \cup R']$ is a minor of G, $\hat{G}[F'' \cup R']$ is also H-minor free. Hence the number of edges in $G[F' \cup R]$ is $O((|F'| + |R|)\,|V(H)|\sqrt{\log|V(H)|})$ [21,29], which is $O(|R|\,|V(H)|\sqrt{\log|V(H)|})$ as desired. If G is planar then the number of edges in $\hat{G}[F'' \cup R']$ is at most $2\,(|F''| + |R'|) \leq 6\,|R'|$ because $\hat{G}[F'' \cup R']$ is a simple planar bipartite graph [30].

Corollary 1. *Node-weighted Steiner tree, Steiner forest, T-join, point-to-point communication, exact tree/cycle/path partitioning problems, lower capacitated partitioning, and location-design and location-routing problems [12] have polynomial-time $O(1)$-approximation algorithms for any family of graphs excluding a fixed minor.*

References

1. I. Abraham, Y. Bartal, and O. Neiman. Nearly tight low stretch spanning trees. In *Proceedings of the 49th Annual Symposium on Foundations of Computer Science*, pages 781–790, 2008.
2. A. Agrawal, P. Klein, and R. Ravi. When trees collide: an approximation algorithm for the generalized Steiner problem on networks. In *STOC*, pages 134–144, 1991.
3. M. Andrews. Hardness of buy-at-bulk network design. In *Proceedings of the 45th Symposium on Foundations of Computer Science*, pages 115–124, 2004.

4. E. Anshelevich, A. Dasgupta, J. Kleinberg, E. Tardos, T. Wexler, and T. Roughgarden. The price of stability for network design with fair cost allocation. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, pages 295–304, 2004.
5. E. M. Arkin and R. Hassin. Approximation algorithms for the geometric covering salesman problem. *Discrete Applied Mathematics*, 55(3):197–218, 1994.
6. Y. Bartal. On approximating arbitrary metrices by tree metrics. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pages 161–168, 1998.
7. D. Bienstock, M. X. Goemans, D. Simchi-Levi, and D. Williamson. A note on the prize collecting traveling salesman problem. *Math. Programming*, 59(3, Ser. A):413–420, 1993.
8. C. Chekuri, S. Khanna, and F. B. Shepherd. Multicommodity flow, well-linked terminals, and routing problems. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, pages 183–192, 2005.
9. U. Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.
10. U. Feige, M. Hajiaghayi, and J. R. Lee. Improved approximation algorithms for minimum-weight vertex separators. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, pages 563–572, 2005.
11. N. Garg, G. Konjevod, and R. Ravi. A polylogarithmic approximation algorithm for the group steiner tree problem. *Journal of Algorithms*, 37(1):66–84, 2000.
12. M. X. Goemans and D. P. Williamson. A general approximation technique for constrained forest problems. *SIAM J. Comput.*, 24(2):296–317, 1995.
13. M. X. Goemans and D. P. Williamson. The primal-dual method for approximation algorithms and its application to network design problems. In D. S. Hochbaum, editor, *Approximation Algorithms for NP-hard Problems*, chapter 4, pages 144–191. PWS, Boston, 1997.
14. J. Gudmundsson and C. Levcopoulos. A fast approximation algorithm for TSP with neighborhoods. *Nordic Journal of Computing*, 6(4):469–488, Winter 1999.
15. S. Guha, A. Moss, J. Naor, and B. Schieber. Efficient recovery from power outage. In *STOC*, pages 574–582, 1999.
16. M. T. Hajiaghayi and K. Jain. The prize-collecting generalized steiner tree problem via a new approach of primal-dual schema. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithm*, pages 631–640, 2006.
17. M. T. Hajiaghayi, R. D. Kleinberg, T. Leighton, and H. Raecke. Oblivious routing on node-capacitated and directed graphs. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 782–790, Philadelphia, PA, USA, 2005.
18. E. Halperin and R. Krauthgamer. Polylogarithmic inapproximability. In *Proceedings of the the 35th Annual ACM Symposium on Theory of Computing*, pages 585–594, 2003.
19. R. M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations (Proc. Sympos., IBM Thomas J. Watson Res. Center, Yorktown Heights, N.Y., 1972)*, pages 85–103. Plenum, New York, 1972.
20. P. Klein and R. Ravi. A nearly best-possible approximation algorithm for node-weighted Steiner trees. *Journal of Algorithms*, 19(1):104–115, July 1995.
21. A. V. Kostochka. Lower bound of the Hadwiger number of graphs by their average degree. *Combinatorica*, 4(4):307–316, 1984.
22. M. V. Marathe, R. Ravi, R. Sundaram, S. S. Ravi, D. J. Rosenkrantz, and H. B. Hunt, III. Bicriteria network design problems. *J. Algorithms*, 28(1):142–171, 1998.

23. C. S. Mata and J. S. B. Mitchell. Approximation algorithms for geometric tour and network design problems (extended abstract). In *Proceedings of the 11th Annual Symposium on Computational Geometry*, pages 360–369, Vancouver, Canada, 1995.
24. J. S. B. Mitchell. A PTAS for TSP with neighborhoods among fat regions in the plane. In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 11–18, 2007.
25. A. Moss and Y. Rabani. Approximation algorithms for constrained node weighted steiner tree problems. *SIAM Journal on Computing*, 37(2):460–481, 2007.
26. R. Raz and S. Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 475–484, 1997.
27. G. Reich and P. Widmayer. Beyond steiner's problem: a VLSI oriented generalization. In *Proceedings of the 15th International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 196–210, 1990.
28. S. Safra and O. Schwartz. On the complexity of approximating TSP with neighborhoods and related problems. *Computational Complexity*, 14(4):281–307, 2006.
29. A. Thomason. The extremal function for complete minors. *Journal of Combinatorial Theory, Series B*, 81(2):318–338, 2001.
30. D. B. West. *Introduction to Graph Theory.* Prentice Hall Inc., Upper Saddle River, NJ, 1996.

A Noncrossing-Flow Directed Steiner Tree

Our relaxation is related to a standard linear-program relaxation of directed Steiner tree, one based on min-cost flows. For each terminal $t \in T$ and each arc e, there is a variable $f_t(e)$. For each arc e, there is a variable $c(e)$. The linear program is as follows:

$$\begin{aligned}
&\text{minimize} \sum_{e \in E} c(e) \max_t f_t(e) \\
&\text{subject to} \sum_{w: vw \in E} f_t(vw) - \sum_{u: uv \in E} f_t(uv) = \begin{cases} 1 \text{ if } v = t \\ -1 \text{ if } v = s \\ 0 \text{ otherwise} \end{cases} \quad \text{for all } t \in T, v \in V \qquad (4)\\
&\qquad f_t(e) \geq 0 \qquad\qquad \text{for all } t \in T, e \in E,
\end{aligned}$$

(The inner max in the objective function can be removed using auxiliary variables, one for each arc.) We denote an assignment to all the variables $f_t(e)$ by f, and we denote by $c(f)$ the corresponding value of the objective function.

Consider the case where G is a planar embedded graph. We say that two paths P and Q in G *cross* if P enters Q on the left, shares zero or more edges with Q, and then exits Q on the right, or vice versa. For a terminal t, a *flow path for* t is a path consisting of arcs e such that $f_t(e) > 0$. We say that an assignment to the variables $f_t(e)$ of the linear program is *noncrossing* if, for every pair t, t' of distinct terminals, every flow path p for t, and every flow path q for t', p and q do not cross. The *noncrossing-flow formulation* of directed Steiner tree refers to the linear program augmented with the (nonlinear) constraint that the flow assignment is noncrossing.

Any minimal solution to directed Steiner tree is a directed tree, so flow paths in the solution do not cross. It follows that the noncrossing-flow formulation is a relaxation for directed Steiner tree in a planar graph. In particular, the optimum of that noncrossing-flow formulation is a lower bound on the minimum cost of a directed Steiner problem. Theorem 6 provides a converse.