Packing Steiner trees

Kamal Jain
Microsoft Research
One Microsoft Way
Redmond, WA 98052
kamalj@microsoft.com

Mohammad Mahdian *
Lab. for Computer Science
M.I.T.
Cambridge, MA 02139
mahdian@mit.edu

Mohammad R. Salavatipour†
Dept. of Computer Science
University of Toronto
Toronto, ON M5S 3G4
mreza@cs.toronto.edu

Abstract

The Steiner packing problem is to find the maximum number of edge-disjoint subgraphs of a given graph G that connect a given set of required points S. This problem is motivated by practical applications in VLSI-layout and broadcasting, as well as theoretical reasons. In this paper, we study this problem and present an algorithm with an asymptotic approximation factor of $|S|/4$. This gives a sufficient condition for the existence of k edge-disjoint Steiner trees in a graph in terms of the edge-connectivity of the graph. We will show that this condition is the best possible if the number of terminals is 3. At the end, we consider the fractional version of this problem, and observe that it can be reduced to the minimum Steiner tree problem via the ellipsoid algorithm.

1 Introduction

In the *Steiner tree packing* problem the objective is to find the maximum number of edge-disjoint Steiner trees in a given graph (See Section 2 for the definitions). In this paper we study the version in which all the Steiner trees are required to connect the same set of *terminal* vertices (a.k.a. *required vertices*). This problem is motivated by both practical as well as theoretical considerations.

The problem in its full generality (where for each Steiner tree, a different set of terminals is given) has applications in VLSI circuit design [15, 9, 8]. In this application, a Steiner tree is needed to share an electric signal by a set of terminal nodes. Another application, which is also our primary focus, arises in the Internet domain. Imagine that a given graph represents a network. Suppose one of the node in the graph is the *broadcaster*. All other nodes are either *users* or *routers* (also called *switches*). The broadcaster wants to broadcast as many streams of movies as possible, so that the users have the maximum number of choices. Each stream of movie is broadcasted via a Steiner tree connecting all the users with the broadcaster. Since we allow parallel edges, we can also assume that each link can carry only one broadcast. So in essence we need to find the maximum number of edge-disjoint Steiner trees connecting all the users and the broadcaster.

From a theoretical perspective, both extremes of this problem are fundamental theorems in combinatorics. One extreme of the problem is when we only have two terminals. In this case a Steiner tree is just a path between the terminals, so the problem becomes the well-known Menger theorem [16]. The other extreme is when all the vertices are terminals. In this case a Steiner tree is just a spanning tree of the graph, so the problem becomes the classical Nash-Williams-Tutte theorem [17, 19].

THEOREM 1.1. *[17, 19] Graph $G(V, E)$ contains k edge-disjoint spanning trees if and only if*

$$E_G(\mathcal{P}) \geq k(t-1)$$

for every partition $\mathcal{P} = \{V_1, \ldots, V_t\}$ of V into non-empty subsets, where $E_G(\mathcal{P})$ denotes the number of edges between distinct classes of $\mathcal{P}$.

Since the problem of finding k disjoint spanning trees in a graph is a special case of finding k disjoint bases of a matroid, Theorem 1.1 can be derived from Edmonds' matroid partition theorem [6].

Both these theorems, Menger as well as Nash-Williams-Tutte theorem, are max-min theorems and can be generalized into a single theorem which says that the

*Research supported by Microsoft Research Theory Group, as part of the summer internship program

†Supported by Research Assistantship, Department of Computer Science, and University open fellowship, University of Toronto.

maximum number of edge-disjoint Steiner trees is the same as the integer part of the minimum value of $\frac{E_G(\mathcal{P})}{|\mathcal{P}|-1}$, where the minimum is over the set of all partitions of the vertices of graph that include at least one terminal in each class, with $E_G(\mathcal{P})$ denoting the number of edges between distinct classes of $\mathcal{P}$, and $|\mathcal{P}|$ denoting the number of classes of $\mathcal{P}$. This statement is not true if we are not in the extreme cases, as shown by an example in [14].

It is an easy exercise to show using Theorem 1.1 that if a graph G is $2k$-edge-connected then it has k edge-disjoint spanning trees. Kriesell [14] conjectured that this corollary generalizes to Steiner trees, i.e., if a set S of vertices of G is $2k$-edge-connected (see Section 2 for the definitions) then there is a set of k edge-disjoint S-Steiner trees in G. This conjecture is still open, even with $2k$ replaced by any constant multiple of k. Notice that the edge-connectivity of the set S is an upper bound on the maximum number of edge-disjoint S-Steiner trees. Thus, a constructive proof for the above conjecture would provide a constant-factor approximation algorithm for the Steiner tree packing problem.

The special case in which $V - S$ is independent is considered by Kriesell [14] and Frank et al. [7]. A corollary of the main theorem of [14] is that for a graph $G(V, E)$ and $S \subseteq V$, if $V - S$ is an independent set and S is $k(k+1)$-edge-connected, then G contains k edge-disjoint S-Steiner trees. A stronger version of this appeared in [7], where they weaken the requirement for connectivity of S to $3k$-edge-connectedness. They also prove a generalization of Theorem 1.1 to hypergraphs.

For an arbitrary set S, Petingi and Rodriguez [18] give a lower bound for the number of edge-disjoint S-Steiner trees, by showing that: if S is k-edge-connected in G and $|S| \geq 2$, then G has at least $\lfloor (\frac{2}{3})^{|V-S|} k/2 \rfloor$ edge-disjoint S-Steiner trees.

For planar graphs, related problems have been considered by Wagner [20]. A variant of the maximum capacity broadcast problem, in which network encoding is allowed is considered in information theory [1, 13].

In this paper, we will present an algorithm for the Steiner tree packing problem which shows that if a subset S in a graph is k-edge-connected, then there are $\alpha_{|S|}k$ edge-disjoint Steiner trees for S, where α_s is a sequence that tends to $\frac{4}{s}$ as s tends to infinity. This result is tight when S consists of 3 points. The novel idea of the algorithm is in trying to combine a collection of edge-disjoint paths with a collection of edge-disjoint Steiner trees by modifying the paths and trees in several (polynomial) iterations.

We also study the problem of packing the maximum number of Steiner trees fractionally. We show that finding an α-approximation algorithm for this problem is equivalent to finding an α-approximation algorithm for the minimum Steiner tree problem. As a corollary, we get a 1.598-approximation algorithm for the fractional Steiner tree packing problem. This also shows that it is hard to find a PTAS for (fractional or integral) Steiner tree packing problem.

2 Preliminaries

We only consider undirected connected finite graphs. A *path* between two vertices u and v in a graph G is a sequence $u = v_0, e_1, v_1, e_2, v_2, \ldots, e_k, v_k = v$ such that e_i's are distinct edges of G, and the endpoints of e_i are v_{i-1} and v_i. Such a path is called *simple* if v_i's are all distinct. The *edge-connectivity* of a connected graph G is the minimum number k of edges required to remove from G to make it disconnected. Let $G(V, E)$ be a graph and $S \subseteq V$ be a set of at least two vertices. We say S is k-edge-connected in G if for any set F of less than k edges in G, there is a path between any pair of vertices in S in $G \setminus F$. In other words, the S-edge-connectivity of G is the minimum number of edges whose removal disconnects at least two vertices of S.

For a graph $G(V, E)$ and a set $S \subseteq V$ of at least two vertices, an S-Steiner tree is a subgraph $T(V', E')$ of G which is a tree and $S \subseteq V'$. The *minimum Steiner tree problem* is the problem of finding the minimum weight S-Steiner tree for a given weighted graph G and subset S of its vertices. The *Steiner tree packing problem* for a given graph $G(V, E)$ and $S \subseteq V$ asks to find a set of maximum size of edge-disjoint S-Steiner trees of G.

3 The Algorithm

In this section we show that for a graph $G(V, E)$ and $S \subseteq V$ with $|S| = s$, if S is k-edge-connected, then there are $\alpha_s k$ edge-disjoint Steiner trees for S, where α_s is a sequence that is asymptotic to $\frac{4}{s}$ as s tends to infinity. For simplicity of exposition, we first prove the following weaker theorem, whose proof contains the main idea that we will use later on in proving the above result.

THEOREM 3.1. *Let $G(V, E)$ be a graph and $S = \{v_1, v_2, v_3\}$ be a subset of V. Assume that v_1 and v_2 are k-edge-connected, and v_1 and v_3 are $\frac{k}{2}$-edge-connected in G. Then G has $\frac{k}{2}$ edge-disjoint S-Steiner trees.*

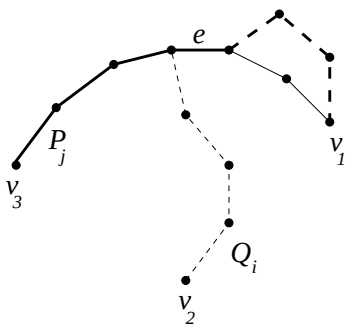

Figure 1: Shortcutting procedure. (The path P_j^* is marked by thick lines).

Proof. Let $\mathcal{P} = \{P_1, P_2, \ldots, P_{\frac{k}{2}}\}$ be a set of $\frac{k}{2}$ edge-disjoint paths between v_1 and v_3. Similarly, let $\mathcal{Q} = \{Q_1, Q_2, \ldots, Q_k\}$ be a set of k edge-disjoint paths between v_1 and v_2. We consider the paths of $\mathcal{P}$ *from* v_3 *to* v_1; thus, an edge e appears *before* another edge e' on P_i if e is closer to v_3 than e' on P_i. Similarly, we consider the paths of $\mathcal{Q}$ from v_2 to v_1.

Note that the paths in $\mathcal{P}$ are not necessarily disjoint from the paths in $\mathcal{Q}$. The *last* intersection of a path Q_i with $\mathcal{P}$ is the last edge (i.e., the edge closest to v_1) on Q_i that is also part of a path in $\mathcal{P}$.

Assume that there exists a path $Q_i \in \mathcal{Q}$ whose last intersection with $\mathcal{P}$ is $e \in P_j$ and e is not the last edge of P_j. We call such a situation a *wasteful* situation, in which we perform the following procedure, called the *shortcutting* procedure: we construct a new path P_j^*, by replacing the part of P_j *after* e with the part of Q_i after e (See Figure 1 for an example). Notice that depending on whether e is traversed by P_j and Q_i in the same or opposite directions, we will have to include or not include e in P_j^*.

Since e is the last intersection of Q_i with $\mathcal{P}$, therefore P_j^* does not intersect any path in $\mathcal{P} \setminus P_j$. Now we remove P_j from $\mathcal{P}$ and instead, add P_j^* to it. After doing this, $\mathcal{P}$ is sill a set of $k/2$ edge-disjoint paths from v_3 to v_1. We call this a *shortcutting* of P_j on Q_i, and say that Q_i is that path that is used for shortcutting P_j^*.

We keep doing the shortcutting procedure, as long as we are in a wasteful situation. Let $\mathcal{Q}^*$ denote the set of paths in $\mathcal{Q}$ that are used for shortcutting a path in $\mathcal{P}$ ($\mathcal{Q}^*$ changes in each iteration of the algorithm). We notice that at any time each path in $\mathcal{P}$ can be shortcut through at most one path in $\mathcal{Q}$; therefore, if P_j is first shortcut through Q_i (i.e., it is replaced by the path P_j^* constructed above) and at a later iteration, P_j^* is shortcut through $Q_{i'}$, then at this iteration we remove i from $\mathcal{Q}^*$ and add i' instead. Let T denote the number of pairs (i, j) such that $P_j \in \mathcal{P}$ and $Q_i \in \mathcal{Q}$ intersect (T changes in each iteration of the algorithm). It is not difficult to observe the following facts.

FACT 3.1. *The shortcutting procedure never increases the value of T.*

FACT 3.2. *Each iteration of the above algorithm either increases the number paths in $\mathcal{Q}^*$, or decreases T.*

It follows immediately from the above facts that the algorithm eventually stops in a non-wasteful situation. In such a situation, every path $Q_i \in \mathcal{Q}$ that has a non-empty intersection with $\mathcal{P}$ is used for shortcutting one path in $\mathcal{P}$ (i.e., belongs to $\mathcal{Q}^*$). Therefore, the number of paths in $\mathcal{Q}$ that have an intersection with a path in $\mathcal{P}$ is not more than $k/2$. So if we remove all the edges of the paths in $\mathcal{P}$ from G, there are still $\frac{k}{2}$ edge-disjoint paths from v_1 to v_2. Each such path together with a path from $\mathcal{P}$ forms an S-Steiner tree. □

By induction on S and using the shortcutting procedure, it can be proved that for a graph $G(V, E)$ and $S \subseteq V$ where $S = \{v_1, v_2, \ldots, v_s\}$, if v_1 and v_i are $(i-1)k$-edge-connected in G, for $2 \leq i \leq s$, then there are k edge-disjoint S-Steiner trees in G. We can guarantee the existence of the same number of edge-disjoint S-Steiner trees, under the slightly stronger assumption that S is $(s-1)k$-edge-connected, using the following simple argument: add a new vertex u and connect it to each of $v_2, v_3 \ldots, v_s$ with k parallel edges. It is easy to see that in this graph, u and v_1 are $(s-1)k$-edge-connected and therefore, by Menger's theorem, there are $(s-1)k$ edge-disjoint paths from u to v_1. These paths can be partitioned into $s-1$ groups, such that the ith group consists of k paths between v_{i+1} and v_1. It is easy to combine these paths to obtain k edge-disjoint S-Steiner trees in G.

There are a few points worth mentioning here. Firstly, as you may have noticed, both of these results guarantee the existence of a collection of Steiner trees all of which are *stars*. This is stronger than what we need, and of course, it doesn't come for free: we need relatively strong assumptions. Secondly, there is an interesting connection between the algorithm in the proof of Theorem 3.1 and the standard stable marriage algorithm. See Conforti et al. [5] for more details.

In the next theorem, we can get a better bound using the idea used in the proof of Theorem 3.1.

THEOREM 3.2. *Let $G(V,E)$ be a graph and S be a subset of s vertices of G. If S is k-edge-connected in G, then there are $\lfloor \alpha_s k \rfloor$ edge-disjoint S-Steiner trees in G, where α_i is defined by the following recurrence relation.*

$$\alpha_2 = 1 \qquad \forall i > 2, \quad \alpha_i = \alpha_{i-1} - \alpha_{i-1}^2/4 \tag{3.1}$$

Proof Sketch: We use induction on s. If $s = 2$, the theorem follows from Menger's theorem. Suppose $s > 2$, and let $v_1, v_2, \ldots, v_s$ be the vertices in S. Define $S' := S \setminus \{v_s\}$. By induction hypothesis, there is a collection $\mathcal{T}$ of $\alpha_{s-1}k$ edge-disjoint S'-Steiner trees in G. We denote these Steiner trees by $T_1, \ldots, T_{\alpha_{s-1}k}$. Also, from Menger's theorem, we know there are k edge-disjoint paths $P_1, P_2, \ldots, P_k$ between v_s and v_1. We consider these paths as paths starting from v_s. So, we say an edge e appears before (after) e' in P_j if e is closer (farther) to v_s than e'.

The basic idea is to combine these trees and paths to obtain Steiner trees for S. The difficulty arises when there trees and paths have some edges in common. We say that an edge e is a *red* edge if it is both in a tree T_i and a path P_j. Consider a tree, say T_1. For each required point v_i, $1 \le i \le s-1$, find the closest red edge to v_i in T_1. Let e be a red edge in T_1, i.e., e is in $T_1 \cap P_j$ for some path P_j. If e is the closest red edge to several vertices $v_{i_1}, v_{i_2}, \ldots, v_{i_\ell}$ in T_1, then we "shortcut" the path P_j to $v_{i_1}, v_{i_2}, \ldots, v_{i_\ell}$ at e. That is, we remove the part of P_j after e, and add to it the paths in T_1 between e and $v_{i_1}, v_{i_2}, \ldots, v_{i_\ell}$. Notice that after this operation, P_j is no longer a path; it starts from v_s as a path, but after reaching e it branches into several branches each ending in one of v_{i_r}'s. We call such a structure a *path-tree*, as we want to emphasize the distinction between the part between v_s and e (which comes from the original path P_j), and the part after e (that comes from the tree T_1).

It is not difficult to see that the collection of P_j's after the above shortcutting procedure can be assumed to be edge-disjoint, without loss of generality, since an intersection between P_j's can occur after the above procedure only if the paths between two required points v_i and v_j and their respective closest red edges e_i and e_j intersect, and in such a situation we can pick e_i as the closest red edge to both v_i and v_j (See Figure 2).

After we shortcut all paths that intersect T_1 in a red edge that is the closest red edge to one of v_i's, we perform the same procedure for T_2. However, if a path P_j is shortcut at e while processing T_1, the edges of P_j that are discarded in this process (i.e., edges that come after e in P_j), are no longer considered red edges. We perform the shortcutting procedure on all trees $T_1, \ldots, T_{\alpha_{s-1}k}$.

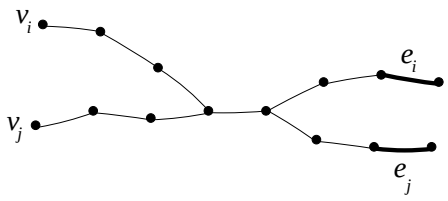

Figure 2: Proof of Theorem 3.2

After this process, we might have paths that are shortcut twice. For example, P_j might be shortcut at an edge e while processing T_1, and at an edge e' before e while processing T_2. If such a thing happens, we mark the edges of the part of P_j that is discarded during the shortcutting procedure for T_2 (i.e., the edges that come after e') as non-red edges (so e is no longer a red edge), and perform the procedure for T_1 again. That is, each of the vertices $v_{i_1}, v_{i_2}, \ldots, v_{i_\ell}$ that had e as their closest red edge in T_1 previously will have to choose their closest red edge again, with the updated set of red edges. We iterate this procedure until every P_j is shortcut for at most one tree. It is not difficult to see that this procedure ends, since in every iteration some of the edges that were originally in P_j are discarded.

At the end of the above procedure, we are in a situation where each P_j is shortcut for at most one tree T_i. Let $\mathcal{Q}_s$ denote the collection of P_j's that are not shortcut (and therefore are still paths from v_s to v_1), let f denote the size of $\mathcal{Q}_s$. Also, let x_i $(i = 0, \ldots, s-1)$ denote the number of T_j's that are used for shortcutting exactly i paths. For $i > 0$, from these trees and the paths that are shortcut through them, we get a collection of ix_i path-trees, that we denote by $\mathcal{Q}_i$. We let $\mathcal{Q}_0$ denote the collection of x_0 trees that are not used in shortcutting any path.

From the above definitions, since there are $k - f$ of P_j's that are shortcut exactly once in the above procedure, we have

$$\sum_{i=0}^{s-1} i x_i = k - f.$$

For simplicity, we let $x_s := f/s$. Therefore, the above equation can be written as

$$\sum_{i=0}^{s} i x_i = k. \tag{3.2}$$

Also, since each of the T_j's is counted in exactly one of x_i's, we have

$$\sum_{i=0}^{s-1} x_i = \alpha_{s-1} k. \tag{3.3}$$

It is not hard to see that the collection of all k paths and path-trees in $\mathcal{Q}_1 \cup \cdots \cup \mathcal{Q}_s$ and x_0 trees in $\mathcal{Q}_0$ constitute a collection $\mathcal{Q}$ of $k + x_0$ edge-disjoint subgraphs of G. In the rest of the proof, we will try to combine the subgraphs in $\mathcal{Q}$ to construct edge-disjoint S-Steiner trees in G.

Let p be a number such that $\sum_{i=p+1}^{s} ix_i < x_0 \leq \sum_{i=p}^{s} ix_i$. If $x_0 \leq sx_s$, we define $p = s$, and if $\sum_{i=1}^{s} ix_i < x_0$, we define $p = 0$. For every $i = p+1, \ldots, s-1$, from each of the ix_i path-trees in $\mathcal{Q}_i$, we pick one path from v_s to one of $v_1, \ldots, v_{s-1}$. Also, $\mathcal{Q}_s$ is by itself a collection of sx_s paths from v_s to v_1. Thus, we can obtain $\sum_{i=p+1}^{s} ix_i$ edge-disjoint paths from v_s to one of $v_1, \ldots, v_{s-1}$, from $\mathcal{Q}_{p+1} \cup \cdots \cup \mathcal{Q}_s$. There are px_p path-trees in $\mathcal{Q}_p$, corresponding to x_p trees in $\mathcal{T}$. Consider the path-trees corresponding to $\lceil (x_0 - \sum_{i=p+1}^{s} ix_i)/p \rceil$ of these trees, and from each of these $p\lceil (x_0 - \sum_{i=p+1}^{s} ix_i)/p \rceil$ path-trees, take one path from v_s to one of $v_1, \ldots, v_{s-1}$. This gives us a collection of $p\lceil (x_0 - \sum_{i=p+1}^{s} ix_i)/p \rceil \geq x_0 - \sum_{i=p+1}^{s} ix_i$ paths from v_s to one of $v_1, \ldots, v_{s-1}$. Therefore, we get at least x_0 edge-disjoint paths from v_s to one of $v_1, \ldots, v_{s-1}$ at the expense of destroying the path-trees in $\mathcal{Q}_s, \mathcal{Q}_{s-1}, \ldots, \mathcal{Q}_{p+1}$, and the path-trees in $\mathcal{Q}_p$ corresponding to $\lceil (x_0 - \sum_{i=p+1}^{s} ix_i)/p \rceil$ trees in $\mathcal{T}$. Each of these paths can be joined with one of the trees in $\mathcal{Q}_0$ to form an S-Steiner tree. The remaining $px_p - p\lceil (x_0 - \sum_{i=p+1}^{s} ix_i)/p \rceil$ path-trees in $\mathcal{Q}_p$ can be grouped into $x_p - \lceil (x_0 - \sum_{i=p+1}^{s} ix_i)/p \rceil$ groups, each group consisting of p path-trees that correspond to the same tree in $\mathcal{T}$. The union of the path-trees in each group is a graph that connects all vertices in S, and therefore contains an S-Steiner tree. This gives us $x_p - \lceil (x_0 - \sum_{i=p+1}^{s} ix_i)/p \rceil$ S-Steiner trees. Similarly, from each $\mathcal{Q}_i$, $i = p, p-1, \ldots, 1$, we get x_i S-Steiner trees. Therefore, the total number of S-Steiner trees that we obtain is equal to

$$\begin{aligned} SOL_p(x) &= x_0 + x_p - \lceil (x_0 - \sum_{i=p+1}^{s} ix_i)/p \rceil + \sum_{i=1}^{p-1} x_i \\ &= \left\lfloor \frac{p-1}{p} x_0 + \sum_{i=1}^{p} x_i + \sum_{i=p+1}^{s} \frac{i}{p} x_i \right\rfloor \end{aligned} \tag{3.4}$$

Here we have to be careful about the two special cases $p = 0$ and $p = s$. Using the same method, it is not difficult to see that in these two cases we get $SOL_0(x) = \sum_{i=1}^{s} ix_i$ and $SOL_s(x) = \sum_{i=0}^{s-1} x_i$ S-Steiner trees, respectively. By Equations (3.2) and (3.3), we have $SOL_0(x) = k$ and $SOL_s(x) = \alpha_{s-1}k$. Therefore, in these cases we get at least $\alpha_{s-1}k > \alpha_s k$ edge-disjoint S-Steiner trees. Thus, we may assume without loss of generality that $1 \leq p < s$.

Now that we have computed the number of S-Steiner trees that our algorithm finds in terms of x_i's, we can analyze the worst-case behavior of our algorithm by treating x_i's as variables and solving the following linear program.

$$\begin{aligned} \text{minimize} \quad & \frac{p-1}{p} x_0 + \sum_{i=1}^{p} x_i + \sum_{i=p+1}^{s} \frac{i}{p} x_i \\ \text{subject to} \quad & \sum_{i=0}^{s} ix_i = k \\ & \sum_{i=0}^{s-1} x_i = \alpha_{s-1} k \\ & \forall i: \quad x_i \geq 0 \end{aligned} \tag{3.5}$$

This is very similar to the idea of using factor-revealing LP's explained in [12, 11]. In order to upper bound the solution of the above linear program, we multiply its first constraint by $1/p^2$ and its second constraint by $(p-1)/p$. We obtain the following.

$$\begin{aligned} \sum_{i=0}^{s-1} \left(\frac{p-1}{p} + \frac{i}{p^2} \right) x_i + \frac{s}{p^2} x_s = \\ \left(\frac{1}{p^2} + \frac{p-1}{p} \alpha_{s-1} \right) k \end{aligned} \tag{3.6}$$

It is easy to see that for $i \leq p$, $\frac{p-1}{p} + \frac{i}{p^2} \leq 1$ and for $i > p$, $\frac{p-1}{p} + \frac{i}{p^2} < \frac{i}{p}$. Also, $\frac{s}{p^2} < \frac{s}{p}$. Thus, since $x_i \geq 0$ for every i,

$$\begin{aligned} \frac{p-1}{p} x_0 + \sum_{i=1}^{p} x_i + \sum_{i=p+1}^{s} \frac{i}{p} x_i \\ \geq \sum_{i=0}^{s-1} \left(\frac{p-1}{p} + \frac{i}{p^2} \right) x_i + \frac{s}{p^2} x_s \\ = \left(\frac{1}{p^2} + \frac{p-1}{p} \alpha_{s-1} \right) k \end{aligned} \tag{3.7}$$

This shows that in the worst case our algorithm finds at least $\lfloor (\frac{1}{p^2} + \frac{p-1}{p} \alpha_{s-1}) k \rfloor$ edge-disjoint S-Steiner trees. The minimum of this expression is at $p = 2/\alpha_{s-1}$. Thus, our algorithm finds is at least $\lfloor (\alpha_{s-1} - \alpha_{s-1}^2/4) k \rfloor = \lfloor \alpha_s k \rfloor$ edge-disjoint S-Steiner trees. □

Notice that the algorithm given in the proof of Theorem 3.2 can be easily implemented in polynomial time. Thus, since the edge-connectivity of the set S is an upper bound on the maximum number of edge-disjoint S-Steiner that we can pack in G, we get the following corollary.

COROLLARY 3.1. *There is a polynomial time algorithm for the Steiner tree packing problem with an approximation ratio of α_s, where s is the number of required points.*

LEMMA 3.1. *Let α_n be the sequence defined by Equation (3.1). Then $\alpha_n = \frac{4}{n} + o(\frac{1}{n})$.*

Proof. Let $\beta_n = \frac{1}{2} - \frac{\alpha_n}{4}$, for $n \geq 2$. Therefore, from Equation (3.1) we have:

$$\beta_n = \beta_{n-1}^2 + \frac{1}{4}. \tag{3.8}$$

CLAIM 3.1. *For $n \geq 2$, $\beta_n \geq \frac{1}{2} - \frac{1}{n}$.*

Proof. We use induction on n. The statement holds trivially for $n = 2$. Suppose $n > 2$ and the claim is true for all values up to $n-1$. By (3.8):

$$\begin{aligned}\beta_n &\geq \left(\frac{1}{2} - \frac{1}{(n-1)}\right)^2 + \frac{1}{4} \\ &= \frac{1}{2} - \frac{n-2}{(n-1)^2} \\ &\geq \frac{1}{2} - \frac{1}{n}.\end{aligned}$$

□

CLAIM 3.2. *For $n \geq 2$, $\beta_n \leq \frac{1}{2} - \frac{1}{4n}$.*

Proof. Again, we use induction on n. The base case $n = 2$ is trivially true. Suppose the statement holds for all values up to $n-1$. By (3.8) and the induction hypothesis,

$$\begin{aligned}\beta_n &\leq \left(\frac{1}{2} - \frac{1}{4(n-1)}\right)^2 + \frac{1}{4} = \frac{1}{2} - \frac{4n-5}{16(n-1)^2} \\ &= \frac{1}{2} - \frac{4n^2-5n}{16n(n-1)^2} \leq \frac{1}{2} - \frac{4n^2-8n+4}{16n(n-1)^2} \\ &= \frac{1}{2} - \frac{1}{4n}.\end{aligned}$$

□

Claim 3.2 will be used to prove the following stronger statement.

CLAIM 3.3. *There is a constant c such that*

$$\beta_n \leq \frac{1}{2} - \frac{1}{n} + \frac{c}{n \ln n}.$$

Proof. For small values of n, the claim is true if we let c to be a large enough constant. Let's assume that n is sufficiently large and that the claim is true for all integers up to $n-1$. From Equation (3.8) we have

$$\begin{aligned}\beta_n &\leq \left(\frac{1}{2} - \frac{\ln(n-1) - c}{(n-1)\ln(n-1)}\right)^2 + \frac{1}{4} \\ &= \frac{1}{2} - \frac{(\ln(n-1) - c)[(n-2)\ln(n-1) + c]}{(n-1)^2 \ln^2(n-1)}.\end{aligned}$$

So, to prove the claim, it is enough to show that,

$$\frac{(\ln(n-1) - c)[(n-2)\ln(n-1) + c]}{(n-1)^2 \ln^2(n-1)} \geq \frac{\ln n - c}{n \ln n},$$

or equivalently,

$$\begin{aligned}&n \ln n (\ln(n-1) - c)[(n-2)\ln(n-1) + c] \\ &-(\ln n - c)(n-1)^2 \ln^2(n-1)] \geq 0.\end{aligned} \tag{3.9}$$

The expansion of the left-hand side of (3.9) is

$$\begin{aligned}&-cn^2 \ln n \ln(n-1) + 3cn \ln n \ln(n-1) - c^2 n \ln n \\ &\quad - \ln n \ln^2(n-1) + cn^2 \ln^2(n-1) \\ &\quad -2cn \ln^2(n-1) + c \ln^2(n-1) \\ &\geq cn^2 \ln(n-1)[\ln(n-1) - \ln n] + cn \ln n \ln(n-1) \\ &\quad -c^2 n \ln n - \ln n \ln^2(n-1) + c \ln^2(n-1) \\ &\geq cn \ln(n-1)[\ln n - 2] - c^2 n \ln n \\ &\quad - \ln n \ln^2(n-1).\end{aligned}$$

Let $c = \frac{3}{4} \ln n_0$ and let n_0 be the smallest integer such that

$$\begin{aligned}&cn_0 \ln(n_0 - 1)[\ln n_0 - 2] - c^2 n_0 \ln n_0 \\ &\qquad - \ln n_0 \ln^2(n_0 - 1) \geq 0.\end{aligned}$$

By this definition, the claim is true for $n \leq n_0$ by Claim 3.2, and for $n > n_0$ by Equation (3.10). □

From Claims 3.1 and 3.3,

$$\frac{4}{n} - \frac{4c}{n \ln n} \leq \alpha_n \leq \frac{4}{n}$$

This completes the proof of the lemma. □

The simplest case of the Steiner tree packing problem after the Menger and Nash-Williams-Tutte theorems is perhaps the case where the number of required points is three ($|S| = 3$). In this case, Theorem 3.2 gives the following corollary.

COROLLARY 3.2. *Let $G(V,E)$ be a graph and S be a subset of 3 vertices of G. If S is k-edge-connected in G, then there are $\lfloor \frac{3}{4}k \rfloor$ edge-disjoint S-Steiner trees in G.*

The following example shows that the constant 3/4 in the above corollary cannot be replaced with any larger constant.

Example. Let G be a graph on s vertices with exactly r parallel edges between each pair of its vertices, and let $S = V(G)$. Clearly, S is k-edge-connected, where $k = (s-1)r$. Since each S-Steiner tree has exactly $s-1$ edges, the maximum number of edge-disjoint S-Steiner trees in G is at most $r\binom{s}{2}/(s-1) = rs/2 = \frac{s}{2(s-1)}k$. In particular, when $s = 3$, the graph does not contain more than $\frac{3}{4}k$ edge-disjoint Steiner trees.

The following question remains open. An affirmative answer to this question would provide a common generalization of Menger's theorem and the corollary of Nash-Williams-Tutte's theorem.

QUESTION 3.1. *Is it true that for every graph G and subset S of vertices of G with $|S| = s$, if S is k-edge-connected in G then there are $\frac{s}{2(s-1)}k$ edge-disjoint S-Steiner trees in G?*

4 Packing Steiner trees fractionally

The *fractional Steiner tree packing* problem can be formulated by the following linear program. In the following $\mathcal{T}$ denotes the collection of all S-Steiner trees in a graph G, and c_e is the (given) *capacity* of the edge e.

$$\begin{array}{lll} & \text{maximize} & \sum_{T\in\mathcal{T}} x_T \\ (4.10) & \text{subject to} & \forall e \in E : \sum_{T:e\in T} x_T \le c_e \\ & & \forall T \in \mathcal{T} : \quad x_T \ge 0 \end{array}$$

This problem is a natural relaxation of the Steiner tree packing problem, and one might hope to get a good upper bound by solving the above linear program. Also, fractional packing of Steiner trees is useful in some broadcasting applications.

The dual of the linear program (4.10) is as follows.

$$\begin{array}{lll} & \text{minimize} & \sum_{e\in E} c_e y_e \\ (4.11) & \text{subject to} & \forall T \in \mathcal{T} : \sum_{e\in T} y_e \ge 1 \\ & & \forall e \in E : \quad y_e \ge 0 \end{array}$$

In other words, the dual LP captures the following problem: Assign non-negative weights to the edges of the graph G in such a way that the minimum weight S-Steiner tree has weight at least 1, and a linear function of the weights of the edges is minimized. The difficulty in solving this linear program arises from the fact that the separation oracle for the above linear program is the Steiner tree problem, and is therefore NP-hard. In this section, we observe that it is possible to use the known approximation algorithms for the Steiner tree problem as an *approximate separation oracle* in the Ellipsoid algorithm to find an approximate fractional packing of Steiner trees. The converse is also true, i.e., an approximation algorithm for the fractional Steiner tree packing problem implies an approximation algorithm for the minimum Steiner tree problem.

THEOREM 4.1. *There is an α-approximation algorithm for the maximum fractional Steiner tree packing problem if and only if there is an α-approximation algorithm for the minimum Steiner tree problem.*

Proof Sketch: Assume there is a polynomial time α-approximation algorithm A for finding the minimum weight Steiner tree in a given weighted graph for a given set of required points. We show that there is a polynomial time α-approximation algorithm for finding the maximum fractional packing of Steiner trees in a given capacitated graph for a given set of required points.

We run the ellipsoid algorithm on the linear program (4.11) using the algorithm A as the separation oracle. More precisely, we add the inequality $\sum_{e\in E} c_e y_e \le R$ to the linear program, and use binary search to find the smallest value of R for which the linear program is feasible. The separation oracle acts as follows: First, it checks the inequality $\sum_{e\in E} c_e y_e \le R$. Next, it runs the algorithm A to find the approximate minimum weight Steiner tree in the graph, using y_e's as the weights of the edges. If the answer that A finds has weight less than 1, then we know that y_e's are not a feasible solution of the linear program (4.11), and the Steiner tree of weight less than 1 gives us a separating hyperplane. If the approximate minimum Steiner tree that A finds has weight at least 1, then we accept y_e's as a feasible solution and therefore the ellipsoid algorithm decides that the linear program is feasible. Of course, since A is just an approximation algorithm, the above conclusion might be incorrect, and the linear program might actually be infeasible. However, since the approximation factor of A is at most α, we know that in this case, αy_e's constitute a feasible solution of

the linear program, with R replaced by αR. Therefore, if R^* is the minimum value of R for which the algorithm decides that the linear program is feasible, then we know that the linear program is infeasible for $R^* - \epsilon$ (where ϵ depends on the precision of the algorithm) , and is feasible for αR^*. Therefore, the optimum solution of the dual program (4.11) is between R^* and αR^*.

The above algorithm computes the approximate value of the solution of the primal program (4.10). In order to compute the actual solution, we use the technique used in [4]. The total number of separating hyperplanes found by the above separation oracle while running the ellipsoid algorithm for $R^* - \epsilon$ is bounded by a polynomial. These separation oracles are enough to show that the solution of the dual program (4.11) is at least R^*. Therefore, if we consider the set of primal variables that correspond to these separating hyperplanes, we get a set of polynomially many primal variables. By LP-duality, if we fix the values of the other variables to 0, the resulting program still has solution at least R^*. However, after fixing the values of other variables to 0 we obtain a polynomial size linear program, which we can solve in polynomial time, and find the optimum solution. By the above argument this optimum solution has value at least R^*. Furthermore, we know that the optimum solution of the dual program (4.11), and therefore the primal program (4.10) is not more than αR^*.

Conversely, assume there is an α-approximation algorithm A for finding the maximum fractional Steiner tree packing in a given capacitated graph with a given set of required points. This means that if we denote the polytope defined by the inequalities of the linear program (4.11) by $\mathcal{P}$, then we can approximately optimize on $\mathcal{P}$ in any given direction. In the polar, this means that there is a procedure that for any given line l, finds (approximately) the first facet of the polar of $\mathcal{P}$ that intersects l. This implies that there is an approximate separation oracle for the polar of $\mathcal{P}$. Using this separation oracle and the method described in the first part of the proof, we can obtain an algorithm that for any given direction, finds the approximate optimum point in the polar of $\mathcal{P}$ along that direction. This means that for $\mathcal{P}$, there is a procedure A' that for any given line l, finds (approximately) the first facet of $\mathcal{P}$ that intersects l. It is not difficult to observe that using A', we can (approximately) solve the minimum Steiner tree problem. Furthermore, the above reduction preserves the approximation factor of the algorithm. □

The above theorem together with the algorithm of Hougardy and Prömel [10] and APX-completeness proof of Bern and Plassmann [3] for the minimum Steiner tree problem implies the following corollaries.

COROLLARY 4.1. *There is a 1.598-approximation algorithm for the fractional Steiner tree packing problem.*

COROLLARY 4.2. *The fractional Steiner tree packing problem is APX-hard.*

COROLLARY 4.3. *The problem of packing the maximum number of edge-disjoint Steiner trees is APX-hard.*

Proof Sketch: It is easy to see that if there is an α-approximation algorithm for the Steiner tree packing problem, then by replacing each edge by several parallel edges, one can obtain an $(\alpha - \epsilon)$-approximation algorithm for the fractional Steiner tree packing problem, for any $\epsilon > 0$. □

It worths mentioning that using Mader's splitting-off lemma (see [2]) one can obtain a combinatorial 2-approximation algorithm for the fractional Steiner tree packing problem. The idea is to replace each edge by two parallel edges, and perform the splitting-off procedure on the Steiner points.

5 Conclusion

In this paper, we considered the problem of packing Steiner trees. This problem is a common generalization of both Nash-Williams-Tutte's and Menger's theorems. We gave a polynomial time algorithm that for a graph $G(V, E)$ and a k-edge-connected subset S of vertices of G, it finds $\alpha_s k$-edge-disjoint S-Steiner trees, where $\alpha_{|S|} = 4/|S| + o(|S|^{-1})$. We showed that this is tight for the case $|S| = 3$. There are several problems left open. The first one is to improve the main result of section 3. Ultimately, we would like to answer Question 3.1. An affirmative answer to this question will probably provide a 2-approximation algorithm for the Steiner tree packing problem. As noted in [14], we don't even know whether there is a constant k such that if S is k-edge-connected in G, then there are *two* edge-disjoint S-Steiner trees in G.

Acknowledgment. We would like to thank Santosh Vempala for pointing out the technique used in [4] for finding the solution of an exponential sized linear program from its dual. Many thanks to Laci Lovasz, for his helpful comments, especially for the proof of Theorem 4.1. We would also like to thank Michel Goemans and anonymous referees for their helpful comments about Mader's theorem and the connection between the short-cutting procedure in Theorem 3.1 and the stable marriage algorithm.

References

[1] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung. Network information flow. *IEEE Transactions on Information Theory*, 46(4):1204–1216, 2000.

[2] J. Bang-Jensen, A. Frank, and B. Jackson. Preserving and increasing local edge-connectivity in mixed graphs. *SIAM J. Discrete Math.*, 8(2):155–178, 1995.

[3] M. Bern and P. Plassmann. The Steiner problem with edge lengths 1 and 2. *Information Processing Letters*, 32(4):171–176, 1989.

[4] B. Carr and S. Vempala. Randomized meta-rounding. *Proc. of the 32nd ACM Symposium on the theory of computing (STOC '00)*, 2000.

[5] M. Conforti, R. Hassin, and R. Ravi. Reconstructing flow paths. Unpublished manuscript, available online at R. Ravi's homepage, August 1999.

[6] J. Edmonds. Minimum partition of a matroid into independent sets. *J. Res. Nat. Bur. Standards Sect.*, 869:67–72, 1965.

[7] A. Frank, T. Király, and M. Kriesell. On decomposing a hypergraph into k connected sub-hypergraphs. *Technical report published by the Egreváry Research Group, Budapest, Hungary. ISSN 1587-4451.*, 2001.

[8] M. Grötschel, A. Martin, and R. Weismantel. Packing Steiner Trees: A Cutting Plane Algorithm and Computational Results. *Mathematical Programming*, 72:125–145, 1996.

[9] M. Grötschel, A. Martin, and R. Weismantel. The Steiner Tree Packing Problem in VLSI-Design. *Mathematical Programming*, 78:265 – 281, 1997.

[10] S. Hougardy and H.J. Prömel. A 1.598 Approximation Algorithm for the Steiner Problem in Graphs. *Proc. of 10th ACM-SIAM Symp. on Disc. Alg. (SODA)*, pages 448–453, 1999.

[11] K. Jain, M. Mahdian, E. Markakis, A. Saberi, and V.V. Vazirani. Approximation Algorithms for Facility Location via Dual Fitting with Factor-Revealing LP. unpublished.

[12] K. Jain, M. Mahdian, and A. Saberi. A new greedy approach for facility location problems. In *ACM Symposium on Theory of Computing*, 2002.

[13] R. Koetter and M. Médard. Beyond rounting: An algebraic approach to network coding. to appear in IEEE/ACM Transactions on Networking, 2002.

[14] M. Kriesell. Local spanning trees in graphs and hypergraph decomposition with respect to edge connectivity. Technical Report 257, University of Hannover, 1999.

[15] A. Martin and R. Weismantel. Packing Paths and Steiner Trees: Routing of Electronic Circuits. *CWI Quarterly*, 6:185 – 204, 1993.

[16] K. Menger. Zur allgemeinen Kurventheorie. *Fund. Math.*, 10:95–115, 1927.

[17] C. St. J. A. Nash-Williams. Edge disjoint spanning trees of finite graphs. *J. Lond. Math. Soc.*, 36:445–450, 1961.

[18] L. Petingi and J. Rodriguez. Bounds on the Maximum Number of Edge-disjoint Steiner Trees of a Graph. *Congressus Numerantium*, 145:43–52, 2000.

[19] W. T. Tutte. On the problem of decomposing a graph into n connected factors. *J. Lond. Math. Soc.*, 36:221–230, 1961.

[20] D. Wagner. Simple algorithms for Steiner trees and paths packing problems in planar graphs. *CWI Quarterly*, 6(3):219–240, 1993.