# Improved Approximation Algorithms for (Budgeted) Node-Weighted Steiner Problems

MohammadHossein Bateni[1], MohammadTaghi Hajiaghayi[2,⋆], and Vahid Liaghat[2,⋆]

[1] Google Research, 76 Ninth Avenue, New York, NY 10011
[2] Computer Science Department, Univ of Maryland, A.V.W. Bldg., College Park, MD 20742

**Abstract.** Moss and Rabani [13] study constrained node-weighted Steiner tree problems with two independent weight values associated with each node, namely, cost and prize (or penalty). They give an $O(\log n)$-approximation algorithm for the prize-collecting node-weighted Steiner tree problem (PCST)—where the goal is to minimize the cost of a tree plus the penalty of vertices not covered by the tree. They use the algorithm for PCST to obtain a bicriteria $(2, O(\log n))$-approximation algorithm for the Budgeted node-weighted Steiner tree problem—where the goal is to maximize the prize of a tree with a given budget for its cost. Their solution may cost up to twice the budget, but collects a factor $\Omega(\frac{1}{\log n})$ of the optimal prize. We improve these results from at least two aspects.

Our first main result is a primal-dual $O(\log h)$-approximation algorithm for a more general problem, prize-collecting node-weighted Steiner forest (PCSF), where we have $h$ demands each requesting the connectivity of a pair of vertices. Our algorithm can be seen as a greedy algorithm which reduces the number of demands by choosing a structure with minimum cost-to-reduction ratio. This natural style of argument (also used by Klein and Ravi [11] and Guha et al. [9]) leads to a much simpler algorithm than that of Moss and Rabani [13] for PCST.

Our second main contribution is for the Budgeted node-weighted Steiner tree problem, which is also an improvement to Moss and Rabani [13] and Guha et al. [9]. In the unrooted case, we improve upon an $O(\log^2 n)$-approximation of [9], and present an $O(\log n)$-approximation algorithm without any budget violation. For the rooted case, where a specified vertex has to appear in the solution tree, we improve the bicriteria result of [13] to a bicriteria approximation ratio of $(1 + \epsilon, O(\log n)/\epsilon^2)$ for any positive (possibly subconstant) $\epsilon$. That is, for any permissible budget violation $1 + \epsilon$, we present an algorithm achieving a tradeoff in the guarantee for prize. Indeed, we show that this is almost tight for the natural linear-programming relaxation used by us as well as in [13].

## 1 Introduction

In the rapidly evolving world of telecommunications and internet, design of fast and efficient networks is of utmost importance. It is not surprising, therefore, that the field of network design has continued to be an active area of research since its inception

⋆ Supported in part by NSF CAREER award 1053605, NSF grant CCF-1161626, ONR YIP award N000141110662, DARPA/AFOSR grant FA9550-12-1-0423, and a University of Maryland Research and Scholarship Award (RASA).

F.V. Fomin et al. (Eds.): ICALP 2013, Part I, LNCS 7965, pp. 81–92, 2013.

several decades ago. These problems have applications not only in designing computer and telecommunications networks, but are also essential for other areas such as VLSI design and computational geometry [3]. Besides their appeals in these applications, basic network design problems (such as Steiner Tree, TSP, and their variants) have been the testbed for new ideas and have been instrumental in development of new techniques in the field of approximation algorithms.

In parallel to the study by Moss and Rabani [13], this work focuses on graph-theoretic problems in which two (independent) nonnegative weight functions are associated with the vertices, namely cost $c(v)$ and prize (or penalty) $\pi(v)$ for each vertex $v$ of the given graph $G(V, E)$. The goal is to find a connected subgraph $H$ of $G$ that optimizes a certain objective. We now summarize the four different problems, already introduced in the literature. In the *Net Worth* problem (NW), the goal is to maximize the prize of $H$ minus its cost[1]. It can be proved that this natural problem does not admit any finite approximation algorithm (see the full version of this work). A similar, yet better-known objective is that of minimizing the cost of the subgraph plus the penalty of nodes outside of it (which is called *Prize-Collecting Steiner Tree* (PCST) in the literature). Two other problems arise if one restricts the range of either cost or prize in the desired solution. In particular, the *Quota* problem tries to find the minimum-cost tree among those with a total prize surpassing a given value, whereas the *Budgeted* problem deals with maximizing the prize with a given maximum budget for the cost. The rooted variants ask, in addition, that a certain root vertex be included in the solution. In the $k$-MST problem, the goal is to find a minimum-cost tree with at least $k$ vertices. In the $k$-STEINER TREE problem, given a set of terminals, the goal is to find a minimum-cost tree spanning at least $k$ terminals. We show the following reductions missing from the literature.

**Theorem 1.** *Let $\alpha$, $0 < \alpha < 1$, be a constant. The following statements are equivalent (both for edge-weighted and node-weighted variants):*

- *i There is an $\alpha$-approximation algorithm for the rooted $k$-MST problem.*
- *ii There is an $\alpha$-approximation algorithm for the unrooted $k$-MST problem.*
- *iii There is an $\alpha$-approximation algorithm for the $k$-STEINER TREE problem.*

*Proof.* Here we present the equivalence of (ii) and (iii) (see the full version of this work for that of (i) and (ii)). We note that one way is clear by definition. To prove that (iii) implies (ii), we give a cost-preserving reduction from $k$-STEINER TREE to $k$-MST. Let $< G = (V, E), T, k >$ be an instance of $k$-STEINER TREE with the set of terminals $T \subseteq V$. Let $n = |V|$. For every terminal $v_t \in T$, add $n$ vertices at distance zero of $v_t$. Let $k' = kn + k$ and consider the solution to $k'$-MST on the new graph. Any subtree with at most $k-1$ terminals have at most $(k-1)n + n - 1 = kn - 1$ vertices. Therefore an optimal solution covers at least $k$ terminals. Hence the reduction preserves the cost of optimal solution. □

These results improve the approximation ratio for $k$-Steiner tree. Previously, a 4-approximation algorithm was proved by [14] and a 5-approximation algorithm was due to [4] who had also conjectured the presence of a $2 + \epsilon$-approximation algorithm.

[1] The prize or cost of a subgraph is defined as the total prize or cost of its vertices, respectively

The equivalence of $k$-Steiner tree and $k$-MST combined with the 2-approximation result of Garg [7] leads to a 2-approximation algorithm for $k$-Steiner tree.

A more tractable version of the prize-collecting variant is the edge-weighted case in which the costs (but not the prizes) are associated with edges rather than nodes. The best known approximation ratio for the edge-weighted Steiner tree problem is $1.39$ due to Byrka et al. [5]. For the earlier work on edge-weighted variant we refer the reader to the references of [5]. In this paper, unless otherwise specified all our graphs are node-weighted and undirected.

### 1.1 Contributions and Techniques

*Approximation Algorithm for PCSF.* Klein and Ravi [11] were the first to give an $O(\log h)$-approximation algorithm for the SF problem. Later, Guha et al. [9] improved the analysis of [11] by showing that the approximation ratio of the algorithm of [11] is w.r.t. the fractional optimal solution for the ST problem. The ST problem is a special case of SF where all demands share an endpoint. Very recently and independently of our work, Chekuri et al. [2] give an algorithm with an approximation ratio of $O(\log n)$ w.r.t. to the fractional solution for SF and higher connectivity problems. This immediately provides a reduction from PCSF to the SF problem: one can fractionally solve the LP for PCSF and pay the penalty of every demand for which the fractional solution pays at least half its penalty. Hence, the remaining demands can be (fractionally) satisfied by paying at most twice the optimal solution. Therefore, one can make a new instance of SF with only the remaining demands and get a solution within $O(\log n)$ factor of the optimal solution using the SF algorithm.

We start off by presenting a simple primal-dual $O(\log h)$-approximation algorithm for the node-weighted prize-collecting Steiner forest (PCSF) problem where $h$ is the number of connectivity demands—see Theorem 2. Compared to the PCST algorithm given by Moss and Rabani [13] and Konemann et al. [12], our algorithm for PCSF solves a more general problem and it has a simpler analysis. A reader familiar with the moat-growing framework[2] may recall that algorithms in this framework (e.g., that of Moss and Rabani [13] or Könemann et al. [12]) consist of a *growth phase* and a *pruning phase*. A moat is a set of dual variables corresponding to a laminar set of vertices containing *terminals*—vertices with a positive penalty. The algorithm grows the moats by increasing the dual variables and adding other vertices gradually to guarantee feasibility. In the edge-weighted Steiner tree problem, when two moats collide on an edge, the algorithm buys the path connecting the moats and merges the moats. Roughly speaking, the algorithm stops growing a moat when either it reaches the root, or its total growth reaches the total prize of terminals inside it. This process is not quite enough to obtain a good approximation ratio. At the end of the algorithm we may have paid too much for connecting unnecessary terminals. Thus as a final step one needs to prune the solution in a certain way to obtain the tight approximation ratio of $2 - \frac{1}{n}$.

In the node-weighted problem, one obstacle is that (polynomially) many moats may collide on a vertex. Handling the proper growth of the moats and the process to merge them proves to be very sophisticated. This may have been the reason that for more than

[2] Introduced by Agrawal, Klein, and Ravi (AKR) [1] and Goemans and Williamson (GW) [8].

a decade no one noticed the flaw in the algorithm of Moss and Rabani [13][3]. Indeed the recently proposed algorithm by Könemann et al. [12] is even more sophisticated. In our algorithm, not only do we completely discard the pruning phase, but we also never merge the moats (thus intuitively, a moat forms a disk centered at a terminal). In fact, our algorithm can be thought of as a simple greedy algorithm. Our algorithm runs in iterations, and in each iteration several disks are grown simultaneously on different endpoints of the demands. The growth stops at the largest possible radius where there are no "overlaps" and no disk has run out of "penalty." If the disks corresponding to several endpoints hit each other, a set of paths connecting them is added to the solution and all but one representative endpoint are removed for the next iteration. However, if a disk is running out of penalty, the terminal at its center is removed for the next iteration. The cost incurred at each iteration is a fraction of OPT, proportional to the fraction of endpoints removed, hence the logarithmic term in the guarantee.

Although our primal-dual approach is different from the approach known for SF [11,9], we indeed use the same style of argument to analyze our algorithm. The crux of these algorithms is to reduce the number of components of the solution by using a structure with minimum cost-to-reduction ratio. Besides the simplicity of this trend, it is important that by avoiding the pruning phase, these algorithms may lead to progress in related settings such as streaming and online settings. The moat-growing approach of Konemann et al. [12], however, allows a stronger lagrangian-preserving guarantee [4] for PCST. This property is shown to be quite important for solving various problems such as $k$-MST and $k$-Steiner tree (see e.g. [4,10]).

*Approximation Algorithms for the Budgeted Problem.* Using their algorithm for PCST, Moss and Rabani developed a bicriteria[5] approximation algorithm for the Budgeted problem, one that achieves an approximation factor $O(\log n)$ on prize while violating the budget constraint by no more than factor two [13]. We present in Theorem 3 a modified pruning procedure that improves the bicriteria bound to $(1+\epsilon, O(\log n)/\epsilon^2)$; in other words, if the algorithm is allowed to violate the budget constraint by only a factor $1+\epsilon$ (for any positive $\epsilon$), the approximation guarantee on the prize will be $O(\log n)/\epsilon^2$. In fact, we also show using the natural linear-programming relaxation (used in [13] as well), that it is not possible to improve these bounds significantly—see the full version. In particular, there are instances for which the fractional solution is $\text{OPT}/\epsilon$, however, no solution of cost at most $1+\epsilon$ times the budget has prize more than $O(\text{OPT})$. Our integrality-gap construction fails if the instance is not rooted. Indeed, in that case, we show how to obtain an $O(\log n)$-approximation algorithm with no budget violations—see Theorem 4. This improves the $O(\log^2 n)$-approximation algorithm of Guha et al. [9].[6] To get over the integrality gap of the LP formulation, we prove several

[3] In private correspondence the authors of the original work have admitted that their algorithm is flawed and that it cannot be fixed easily.

[4] Let $T$ denote the sets of vertices purchased by the algorithm of [12]. It is guaranteed that $c(T) + \log(n)\pi(V \backslash T) \leq \log(n)\text{OPT}$.

[5] An $(\alpha, \beta)$-bicriteria approximation algorithm for the Budgeted problem finds a tree with total prize at least $\frac{1}{\beta}$ fraction of that of optimal solution and total cost at most $\alpha$ factor of the budget.

[6] The $O(\log^2 n)$-approximation algorithm can be derived from the results in [9] with some efforts, not as explicitly as cited by Moss and Rabani [13].

structural properties for near-optimal solutions. By restricting the solution to one with these properties, we use a bicriteria approximation algorithm as a black box to find a near-optimal solution. Finally we use a generalization of the trimming method of [9] to avoid violating the budget.

## 2 The Prize-Collecting Steiner Forest Problem

The starting point of the algorithm of Moss and Rabani [13] is a standard LP relaxation for the rooted version. For the Quota and Budgeted problems they show that any (fractional) feasible solution can be approximated by a convex combination of sets of nodes connected (integrally) to the root. Given the support of such a convex combination, it follows from an averaging argument that a proper set can be found. Thus the problem comes down to finding the support of the convex combination. They show that given a black-box algorithm which solves the PCST problem with the approximation factor $O(\log n)$, one can obtain the support in polynomial time.

The main result of this section is a very simple, and maybe more elegant algorithm for the classical problem of PCSF (and thus PCST). As mentioned before, using moats and having a pruning phase lead to the main difficulty in the analysis of previous algorithms. These seem to be a necessary evil for achieving a tight constant approximation factor for the edge-weighted variant. Surprisingly, we show neither is needed in the node-weighted variant. Instead of moats, we use dual *disks* which are centered on a *single* terminal and we do not need a pruning phase.

### 2.1 Preliminaries

Consider a graph $G = (V, E)$ with a node-weight function $c : V \to \mathbb{R}_{\geq 0}$. For a subset $S \subseteq V$, let $c(S) := \sum_{v \in S} c(v)$. In the *Steiner Forest* problem, given a set of demands $\mathcal{L} = \langle (s_1, t_1), \dots, (s_h, t_h) \rangle$, the goal is to find a set of vertices $X$ such that for every demand $i \in [h]$, $s_i$ and $t_i$ are connected in $G[X]$. The vertices $s_i$ and $t_i$ are denoted as the *endpoints* of the demand $i$. In PCSF a penalty (prize) $\pi_i \in \mathbb{R}_{\geq 0}$ is associated with every demand $i \in [h]$. If the endpoints of a demand are not connected in the solution, we need to pay the penalty of the demand. The objective cost of a solution $X \subseteq V$ is

$$\mathbf{PCSF}(X) = c(X) + \sum_{i \in [h]: i \textit{ is not satisfied}} \pi_i.$$

A *terminal* is a vertex which is an endpoint of a demand. Let $\mathcal{T}$ denote the set of terminals. We may assume that the cost of a terminal is zero. We also assume the endpoints of all demands are different[7] (thus $|\mathcal{T}| = 2h$). For a pair of vertices $u$ and $v$ and a cost function $c$, let $d^c(u, v)$ denote the length of the shortest path with respect to $c$ connecting $u$ and $v$, including the cost of endpoints.

For a set of vertices $S$ let $\delta(S)$ denote the set of vertices that are not in $S$ but have neighbors in $S$. A set $S$ *separates* a demand $i$ if exactly one of $s_i$ and $t_i$ is in $S$. Let

[7] Both assumptions are without loss of generality. For every demand $(s_i, t_i)$, attach a new vertex $s^i$ of cost zero to $s_i$ and similarly attach a new vertex $t^i$ of cost zero to $t_i$. Now interpret $i$ as the demand between $s^i$ and $t^i$. The optimal cost does not change.

$\mathcal{S}_i$ denote the collection of sets separating the demand $i$ and let $\mathcal{S} = \bigcup_i \mathcal{S}_i$. For a set $S$, define the penalty of $S$ as half of the total penalty of demands separated by $S$, i.e., $\pi_{\mathcal{L}}(S) = \frac{1}{2}\sum_{i:S\in\mathcal{S}_i} \pi_i$. We may drop the index $\mathcal{L}$ when there is no ambiguity. The PCSF problem can be formulated as the following standard integer program (IP):

$$\begin{array}{ll} \text{Minimize} & \sum_{v\in V\setminus\mathcal{T}} c(v)\mathbf{x}(v) + \sum_{S\in\mathcal{S}} \pi(S)\mathbf{z}(S) \\ \forall i \in [h], S \in \mathcal{S}_i & \sum_{v\in\delta(S)} \mathbf{x}(v) + \sum_{R|S\subseteq R\in\mathcal{S}_i} \mathbf{z}(R) \geq 1 \\ & \mathbf{x}(v), \mathbf{z}(S) \in \{0,1\} \end{array}$$

Given a solution $X \subseteq V$ to the PCSF problem one can easily make a feasible solution $\mathbf{x}$ to the IP with the same objective value as $\mathbf{PCSF}(X)$: since the cost of a terminal is zero, we assume $\mathcal{T} \subseteq X$. For every vertex $v \in X$ set $\mathbf{x}(v) = 1$ and for every connected component $CC$ of $G[X]$ set $\mathbf{z}(V\setminus CC) = 1$. It is also easy to verify since the cost of a terminal is zero, any (integral) feasible solution $\mathbf{x}$ corresponds to a solution $X \subseteq V$ for the PCSF problem with (at most) the same cost. One may relax the IP by allowing assignment of fractional values to the variables. Let OPT denote the objective value of the optimal solution for the relaxed linear program (LP). The following is the dual program $\mathcal{D}$ corresponding to the relaxed LP.

$$\begin{array}{ll} \text{Maximize} & \sum_{S\in\mathcal{S}} \mathbf{y}(S) \\ \forall v \in V & \sum_{S\in\mathcal{S}:v\in\delta(S)} \mathbf{y}(S) \leq c(v) \\ \forall S \in \mathcal{S} & \sum_{S'\subseteq S} \sum_{i:S,S'\in\mathcal{S}_i} \mathbf{y}_i(S') \leq \pi(S) \\ & \mathbf{y}_i(S) \geq 0, \mathbf{y}(S) = \sum_{i:S\in\mathcal{S}_i} \mathbf{y}_i(S) \end{array} \qquad (\mathcal{D})$$

In the case of Steiner tree, the dual variables are defined w.r.t. a set $S$. However, in Steiner forest, the dual variables are in the form $\mathbf{y}_i(S)$, i.e., they are defined based on a demand as well. This has been one source of the complexity of previous primal-dual algorithms for Steiner forest problems. Interestingly, in our approach, we only need to work with a *simplified dual* constructed as follows.

*Cores and Simplified Duals* Let $c$ and $\mathcal{L}$ denote a node-weight function and a set of demands, respectively. Let $Z_c$ denote the set of vertices with zero cost. We note that the terminals are in $Z_c$. A set $C \subseteq V$ is a *core* if $C$ is a connected component of $G[Z_c]$ *and* contains a terminal (i.e., an endpoint of a demand in $\mathcal{L}$). Let $\overline{\mathcal{S}(c,\mathcal{L})}$ be the collection of sets separating one core from the other cores, i.e., a set $S$ is in $\overline{\mathcal{S}(c,\mathcal{L})}$ if $S$ contains a core but has no intersection with other cores. For a set $S \in \overline{\mathcal{S}(c,\mathcal{L})}$, let $\mathbf{core}(S)$ denote the core inside $S$. Note that $\pi_{\mathcal{L}}(S) = \pi_{\mathcal{L}}(\mathbf{core}(S))$. A simplified dual w.r.t. $c$ and $\mathcal{L}$ is the following program $\overline{\mathcal{D}(c,\mathcal{L})}$.

$$\begin{array}{lll} \text{Maximize} & \sum_{S \in \mathcal{S}} \mathbf{y}(S) & (\overline{\mathcal{D}(c,\mathcal{L})}) \\ \forall v \in V & \sum_{S \in \overline{\mathcal{S}(c,\mathcal{L})} : v \in \delta(S)} \mathbf{y}(S) \le c(v) & \text{(C1)} \\ \forall S \in \overline{\mathcal{S}(c,\mathcal{L})} & \sum_{S' : \mathbf{core}(S) \subseteq S' \subseteq S} \mathbf{y}(S') \le \pi_{\mathcal{L}}(S) & \text{(C2)} \\ & \mathbf{y}(S) \ge 0 & \end{array}$$

Observe that $\overline{\mathcal{S}(c,\mathcal{L})} \subseteq \mathcal{S}$. Indeed $\overline{\mathcal{D}(c,\mathcal{L})}$ is the same as $\mathcal{D}$ with only (much) fewer variables. Thus the program $\overline{\mathcal{D}(c,\mathcal{L})}$ is only more restricted than $\mathcal{D}$. In the rest of the paper, unless specified otherwise, by a dual we mean a simplified dual. When clear from the context, we may omit the indices $c$ and $\mathcal{L}$.

*Disks* Consider a Dual Vector $\mathbf{y}$ Initialized to Zero. A *disk of radius $R$ centered at a terminal $t$* is the dual vector obtained from the following process: Initialize the set $S$ to the core containing $t$. Increase $\mathbf{y}(S)$ until for a vertex $u$ the dual constraint C1 becomes tight. Add $u$ to $S$ and repeat with the new $S$. Stop the process when the total growth (i.e., sum of the dual variables) reaches $R$. A disk is *valid* if $\mathbf{y}$ is feasible. In what follows, by a disk we mean a valid disk unless specified otherwise.

A vertex $v$ is *inside* the disk if $d^c(t,v)$ is strictly less than $R$. The *continent* of a disk is the set of vertices inside the disk. Further, we say a vertex $v$ is on the *boundary* of a disk if it is not inside the disk but has a neighbor $u$ such that $d^c(t,u) \le R$. Note that $u$ is not necessary inside the disk. The following facts about a disk of radius $R$ centered at a terminal $t$ can be derived from the definition:

**Fact 1.** *The (dual) objective value of the disk is exactly $R$.*

**Fact 2.** *For every vertex inside the disk, the dual constraint C1 is tight.*

**Fact 3.** *If a set $S$ does not include the center, then $\mathbf{y}(S) = 0$. Further, if $S$ is not a subset of the continent, then $\mathbf{y}(S) = 0$.*

Let $\mathbf{y}_1, \ldots, \mathbf{y}_k$ denote a set of disks. The *union* of the disks is simply a dual vector $\mathbf{y}$ such that $\mathbf{y}(S) = \sum_i \mathbf{y}_i(S)$ for every set $S \subseteq \mathcal{S}$. A set of disks are *non-overlapping* if their union is a feasible dual solution (i.e., both set of constraints C1 and C2 hold). If a vertex $v$ is inside a disk, the corresponding dual constraint is tight. Thus for any set $S$ such that $v \in \delta(S)$, the dual variable $\mathbf{y}(S)$ cannot be increased. On the other hand since the distance between $v$ and the center is *strictly* less than the radius, there exists a set containing $v$ with positive dual value. This observation leads to the following.

**Proposition 1.** *Let $\mathbf{y}$ be the union of a set of non-overlapping disks $\mathbf{y}_1, \ldots, \mathbf{y}_k$. A vertex inside a disk cannot be on the boundary of another disk.*

Proposition 1 implies that in the union of a set of non-overlapping disks, the continents are pairwise far from each other. This intuition leads to the following .[8]

[8] Due to the lack of space, we have omitted some of the proofs throughout the paper. We defer the reader to the full version of this paper for the omitted proofs.

**Lemma 1.** *Suppose $T'$ is a subset of terminals such that the distance between every pair of them is non-zero. Let $R$ denote the maximum radius such that the $|T'|$ disks of radius $R$ centered at terminals in $T'$ are non-overlapping. Consider the union of such disks. Either (i) the constraint C2 is tight for a continent; or (ii) the constraint C1 is tight for a vertex on the boundary of multiple disks.*

The final tool we need for the analysis of the algorithm states a precise relation between the dual variables and the distance of a vertex on the boundary.

**Lemma 2.** *Let $v$ be a vertex on the boundary of a disk $\mathbf{y}$ of radius $R$ centered at a terminal $t$. We have $\sum_{S|v\in\delta(S)} \mathbf{y}(S) = R - (d^c(t,v) - c(v))$.*

### 2.2 An Algorithm for the PCSF Problem

The algorithm finds the solution $X$ iteratively. Let $X_i$ denote the set of vertices bought after iteration $i$ where $X_0$ is the set of terminals. For every $i$, the *modified cost function* $c_i$ is a copy of $c$ induced by setting the cost of vertices in $X_{i-1}$ to zero, i.e., $c_i = c[X_{i-1} \to 0]$. At iteration $i$ there is a set of *active* demands $\mathcal{L}_i \subseteq \mathcal{L}$ and the dual program $\mathcal{D}_i = \overline{\mathcal{D}(c_i, \mathcal{L}_i)}$. The program $\mathcal{D}_i$ is the simplified dual program w.r.t. the modified cost function and the active demands. Note that $\mathcal{D}_i$ is stricter than $\mathcal{D}$, thus the objective value of a feasible solution to $\mathcal{D}_i$ is a lower bound for OPT. The algorithm guarantees that for every $i < j$, $X_i \subseteq X_j$ and $\mathcal{L}_i$ is a superset of $\mathcal{L}_j$.

The algorithm is as follows (see Algorithm 1). We initialize $X_0 = \mathcal{T}$, $c_1 = c$, and $\mathcal{L}_1 = \mathcal{L}$. At iteration $i$, consider the cores formed w.r.t. $c_i$ and $\mathcal{L}_i$. Let $T_i$ denote a set which has exactly one terminal in each core (so the number of cores is $|T_i|$). The algorithm finds the maximum radius $R_i$ such that the $|T_i|$ disks of radius $R_i$ centered at each terminal in $T_i$ are non-overlapping w.r.t. $\mathcal{D}_i$. By Lemma 1 either the constraint C2 is tight for a continent $S$; or the constraint C1 is tight for a vertex $v$ on the boundary of multiple disks. In the former, deactivate every demand with exactly one endpoint in $\mathbf{core}(S)$; pay the penalty of such demands and continue to the next iteration with the remaining active demands. In the latter, let $L_v$ denote the centers of the disks whose boundaries contain $v$. For every terminal $\tau \in L_v$ buy the shortest path w.r.t. $c_i$ connecting $v$ to $\tau$ (and so to the core containing $\tau$). Deactivate a demand if its endpoints are now connected in the solution and continue to the next iteration. The algorithm stops when there is no active demand remaining; in which case it returns the final set of vertices bought by the algorithm.

We bound the objective cost of the algorithm in each iteration separately. The following theorem shows that the fraction of OPT we incur at each iteration is proportional to the reduction in the number of cores after the iteration.

**Theorem 2.** *The approximation ratio of Algorithm 1 is at most $2H_{2h}$ where $H_{2h}$ is the $(2h)^{th}$ harmonic number.*

*Proof.* Observe that at each iteration, a core is a connected component of the solution which contains an endpoint of at least one active demand. We distinguish between two types of iterations: In Type I, Line 8 of Algorithm 1 is executed while in Type II, Line 15 is executed.

**Algorithm 1.** The Prize-Collecting Steiner Forest Algorithm

**Input:** A graph $G = (V, E)$, a set of demands $\mathcal{L}$ with penalties, and a cost function $c$.

1: Initialize $X_0 = \mathcal{T}$, $\mathcal{L}_1 = \mathcal{L}$, $c_1 = c$, and $i = 1$.
2: **while** $|\mathcal{L}_i| > 0$ **do**
3: Set $c_i = c[X_{i-1} \to 0]$ and construct the dual program $\mathcal{D}_i$ with respect to $c_i$ and $\mathcal{L}_i$.
4: Construct $T_i$ by choosing an arbitrary terminal from each core.
5: Let $R_i$ be the maximum radius such that putting a disk of radius $R_i$ centered at every terminal in $T_i$ is feasible w.r.t. $\mathcal{D}_i$.
6: **if** the constraint C2 is tight for a continent $S$ **then**
7: Set $X_i = X_{i-1}$.
8: Set $\mathcal{L}_{i+1} = \mathcal{L}_i \setminus \{j \in [h] | \text{ either } s_j \in \mathbf{core}(S) \text{ or } t_j \in \mathbf{core}(S)\}$.
9: **else**
10: Find a vertex $v$ on the boundary of multiple disks for which constraint C1 is tight.
11: Let $L_v$ denote the centers of the disks whose boundaries contain $v$.
12: Initialize $X_i = X_{i-1}$.
13: **for all** $\tau \in L_v$ **do**
14: Add the shortest path (w.r.t. $c_i$) between $\tau$ and $v$ to $X_i$.
15: Set $\mathcal{L}_{i+1} = \mathcal{L}_i \setminus \{j \in [h] | d^{c_{i+1}}(s_j, t_j) = 0\}$.
16: $i = i + 1$.
17: Output $X_{i-1}$.

Observe that a demand is deactivated either at Line 8 or at Line 15. In the latter, the endpoints of a demand are indeed connected in the solution. Thus we only need to pay the penalty of a demand if it is deactivated in an iteration of Type I. Recall that at Line 8, the penalty of $\mathbf{core}(S)$ is half the total penalty of demands cut by $S$. Thus the total penalty we incur at that line is exactly $2\pi_{\mathcal{L}_i}(S)$

We now break the total objective cost of the algorithm into a payment $P_i$ for each iteration $i$ as follows:

$$P_i = \begin{cases} 2\pi_{\mathcal{L}_i}(S) & \text{for Type I iterations executing Line 8 with the continent } S; \\ c(X_i) - c(X_{i-1}) & \text{for Type II iterations.} \end{cases}$$

Recall that $|T_i|$ is the number of cores at iteration $i$. Observe that by Fact 1, at iteration $i$ the total dual vector has value $R_i|T_i|$. By the weak duality $R_i \leq \frac{\text{OPT}}{|T_i|}$. For every $i \geq 1$, let $h_i = |T_i| - |T_{i+1}|$ denote the reduction in the number of cores after the iteration $i$.

*Claim.* $P_i \leq 2h_iR_i$ for every iteration $i$.

*Proof.* Fix an iteration $i$. Let $\mathbf{y}$ denote the union of disks of radius $R_i$ centered at $T_i$. We distinguish between the two types of the iteration:

- *Type I.* At Line 8, by deactivating all the demands crossing a core, we essentially remove that core. Thus in such an iteration $h_i = 1$. The objective cost of the iteration is $2\pi_{\mathcal{L}_i}(S)$. On the other hand, the constraint C1 is tight for $S$, i.e., $\sum_{S' \subseteq S} \mathbf{y}(S) = \pi_{\mathcal{L}_i}(S)$. By Facts 1 and 3, the radius $R_i$ equals $\sum_{S' \subseteq S} \mathbf{y}(S)$. Therefore the objective cost is at most $2h_iR_i$
- *Type II.* At line 15, we connect $|L_v|$ cores to each other, thus reducing the number of cores in the next iteration by at least $h_i \geq |L_v| - 1$. Recall that by Lemma 1,

$|L_v| \geq 2$ and hence $h_i \geq 1$. The total cost of connecting terminals in $L_v$ to $v$ is bounded by $c_i(v)$ plus for every $\tau \in L_v$, the cost of the path connecting $\tau$ to $v$ excluding $c_i(v)$. Thus $P_i \leq c_i(v) + \sum_{\tau \in L_v}(d^{c_i}(\tau, v) - c_i(v))$. Now we write the equation in Lemma 2 for every disk centered at a terminal in $L_v$:

$$|L_v|R_i = \sum_{\tau \in L_v} [d^{c_i}(\tau, v) - c_i(v) + \sum_{S|v \in \delta(S), \tau \in S} \mathbf{y}(S)]$$
$$= \sum_{\tau \in L_v} [d^{c_i}(\tau, v) - c_i(v)] + c_i(v) \geq P_i,$$

where the last equality follows since the constraint C1 is tight for $v$. Since the disks are non-overlapping, by Fact 3, $\mathbf{y}(S)$ is positive only if it contains a single terminal of $L_v$. This completes the proof since $P_i \leq |L_v|R_i \leq (h_i + 1)R_i \leq 2h_iR_i$. $\square$

Let $X$ be the final solution of the algorithm. Note that $|T_{i+1}| = |T_i| - h_i$ and $|T_1| \leq |\mathcal{T}|$. A simple calculation shows

$$\mathbf{PCSF}(X) \leq \sum_i P_i \leq \sum_i 2h_iR_i \leq 2\text{OPT} \sum_i \frac{h_i}{|T_i|} \leq 2\text{OPT} \cdot H_{|\mathcal{T}|}. \qquad \square$$

## 3 The Budgeted Steiner Tree Problem

In this section we consider the Budgeted problem in the node-weighted Steiner tree setting. Recall that for a vertex $v \in V$, we denote the prize and the cost of the vertex by $\pi(v)$ and $c(v)$, respectively. First we generalize the trimming process of Guha et al. [9] which reduces the budget violation of a solution while preserving the prize-to-cost ratio. We use this process to obtain a bicriteria approximation algorithm for the rooted version in Section 3.1. Next, in Section 3.2 we consider the unrooted version. By providing a structural property of near-optimal solutions, we propose an algorithm which achieves a logarithmic approximation factor without violating the budget constraint; improving on the previous result of Guha et al. [9] which obtains an $O(\log^2 n)$-approximation algorithm without violation.

In what follows, for a rooted tree $T$ we assume a *subtree* rooted at a vertex $v$ consists of all vertices whose path to the root of $T$ passes through $v$. The set of *strict subtrees* of $T$ consists of all subtrees other than $T$ itself. Further, the set of *immediate subtrees* of $T$ are the subtrees rooted at the children of the root of $T$.

### 3.1 The Rooted Budgeted Problem

For a budget value $B$ and a vertex $r$, a graph is *$B$-proper for the vertex $r$* if the cost of reaching any vertex from $r$ is at most $B$. The following lemma shows a bicriteria trimming method (proof in the full version).

**Lemma 3.** *Let $T$ be a subtree rooted at $r$ with the prize-to-cost ratio $\gamma$. Suppose the underlying graph is $B$-proper for $r$ and for $\epsilon \in (0, 1]$ the cost of the tree is at least $\frac{\epsilon B}{2}$. One can find a tree $T^*$ containing $r$ with the prize-to-cost ratio at least $\frac{\epsilon}{4}\gamma$ such that $\frac{\epsilon}{2}B \leq c(T^*) \leq (1 + \epsilon)B$.*

Moss and Rabani [13] give an $O(\log n)$-approximation algorithm for the Budgeted problem which may violate the budget by a factor of two. Using Lemma 3 one can trim such a solution to achieve a trade-off between the violation of budget and the approximation factor (proof in the full version) .

**Theorem 3.** *For every $\epsilon \in (0, 1]$ one can find a subtree $T \subseteq G$ in polynomial time such that $c(T) \leq (1 + \epsilon)B$ and the total prize of $T$ is $\Omega(\frac{\epsilon^2}{\log n})$ fraction of OPT.*

### 3.2 The Unrooted Budgeted Problem

We prove a stronger variant of Lemma 3 for the unrooted version. We show that if no single vertex is too expensive, one does not need to violate the budget at all. The analysis is similar to that of Lemma 3.

**Lemma 4.** *Let $T$ be a tree with the prize to cost ratio $\gamma$. Suppose $c(T) \geq \frac{B}{2}$ and the cost of every vertex of the tree is at most $\frac{B}{2}$ for a real number $B$. One can find a subtree $T^* \subseteq T$ with the prize to cost ratio at least $\frac{\gamma}{4}$ such that $\frac{B}{4} \leq c(T^*) \leq B$.*

One may use arguments similar to that of Theorem 3 to derive an $O(\log n)$-approximation algorithm from Lemma 4 when the cost of a vertex is not too big. On the other hand if the cost of a vertex is more than half the budget, we can guess that vertex and try to solve the problem with the remaining budget. However, one obstacle is that this process may need to be repeated, i.e., the cost of another vertex may be more than half the remaining budget. Thus we may need to continue guessing many vertices in which case connecting them in an optimal manner would not be an easy task. The following theorem shows indeed guessing one vertex is sufficient if one is willing to lose an extra factor of two in the approximation guarantee.

**Theorem 4.** *The unrooted budgeted problem admits an $O(\log n)$-approximation algorithm which does not violate the budget constraint.*

*Proof.* We define two classes of subtrees: the *flat* trees and the *saddled* trees. A tree is flat if the cost of every vertex of the tree is at most $\frac{B}{2}$. For a tree $T$, let $x$ be the vertex of $T$ with the largest cost. The tree $T$ is saddled if $c(x) > \frac{B}{2}$ and the cost of every other vertex of the tree is at most $\frac{B-c(x)}{2}$. Let $T_f^*$ denote the optimal flat tree, i.e., a flat tree with the maximum prize among all the flat trees with the total cost at most $B$. Similarly, let $T_s^*$ denote the optimal saddled tree.

The proof is described in two parts. First we show the prize of the best solution between $T_f^*$ and $T_s^*$ is indeed in a constant factor of OPT(see the following claim, proof in the full version). Next, we show by restricting the optimum to any of the two classes, an $O(\log(n))$-approximation solution can be found in polynomial time. Therefore this would give us the desired approximation algorithm.

*Claim.* Either $\pi(T_f^*) \geq \frac{\text{OPT}}{2}$ or $\pi(T_s^*) \geq \frac{\text{OPT}}{2}$.

Now we only need to restrict the algorithm to flat trees and saddled trees. Indeed we can reduce the case of saddled trees to flat trees. We simply guess the maximum-cost vertex $x$ (by iterating over all vertices). We form a new instance of the problem by reducing the budget to $B - c(x)$ and the cost of $x$ to zero. The cost of every other vertex in $T_s^*$ is at most half the remaining budget, thus we need to look for the best flat tree in the new instance. Therefore it only remains to find an approximation solution when restricted to flat trees.

We use Lemma 4 to find the desired solution for flat trees. A vertex with cost more than half the budget cannot be in a flat tree, thus we remove all such vertices. We may guess a vertex of the best solution and by using the algorithm of Moss and Rabani [13] we can find an $O(\log n)$-approximation solution which may use twice the budget. Let $T$ be the resulting tree with the total prize $P$. If $c(T) \leq B$ we are done. Otherwise by Lemma 4 we can trim $T$ to obtain a tree with the cost at most $B$ and the prize at least $\frac{P}{32}$ which completes the proof. □

## References


1. Agrawal, A., Klein, P., Ravi, R.: When trees collide: an approximation algorithm for the generalized Steiner problem on networks. In: STOC (1991)
2. Chekuri, C., Ene, A., Vakilian, A.: Prize-collecting survivable network design in node-weighted graphs. In: Gupta, A., Jansen, K., Rolim, J., Servedio, R. (eds.) APPROX/RANDOM 2012. LNCS, vol. 7408, pp. 98–109. Springer, Heidelberg (2012)
3. Cheng, X., Li, Y., Du, D.-Z., Ngo, H.Q.: Steiner trees in industry. In: Handbook of Combinatorial Optimization (2005)
4. Chudak, F.A., Roughgarden, T., Williamson, D.P.: Approximate k-MSTs and k-Steiner trees via the primal-dual method and Lagrangean relaxation. Mathematical Programming 100, 411–421 (2004)
5. Erlebach, T., Grant, T., Kammer, F.: Maximising lifetime for fault-tolerant target coverage in sensor networks. In: SPAA (2011)
6. Feigenbaum, J., Papadimitriou, C.H., Shenker, S.: Sharing the cost of multicast transmissions. Journal of Computer and System Sciences 63, 21–41 (2001)
7. Garg, N.: Saving an epsilon: a 2-approximation for the $k$-MST problem in graphs. In: STOC (2005)
8. Goemans, M., Williamson, D.P.: A general approximation technique for constrained forest problems. SIAM J. on Computing 24, 296–317 (1992)
9. Guha, S., Moss, A., Naor, J(S.), Schieber, B.: Efficient recovery from power outage. In: STOC (1999)
10. Jain, K., Vazirani, V.V.: Approximation algorithms for metric facility location and k-Median problems using the primal-dual schema and Lagrangian relaxation. J. ACM
11. Klein, P., Ravi, R.: A nearly best-possible approximation algorithm for node-weighted Steiner trees. J. Algorithms 19(1), 104–115 (1995)
12. Könemann, J., Sadeghian, S., Sanita, L.: An LMP $O(\log n)$-approximation algorithm for node weighted prize collecting Steiner tree (unpublished, 2013)
13. Moss, A., Rabani, Y.: Approximation algorithms for constrained node weighted Steiner tree problems. SIAM J. Comput. 37(2), 460–481 (2007)
14. Ravi, R., Sundaram, R., Marathe, M.V., Rosenkrantz, D.J., Ravi, S.S.: Spanning trees - short or small. SIAM J. Discrete Math. 9(2), 178–200 (1996)