SIAM J. COMPUT.
Vol. 47, No. 4, pp. 1275–1293

IMPROVED APPROXIMATION ALGORITHMS FOR (BUDGETED) NODE-WEIGHTED STEINER PROBLEMS*

MOHAMMAD HOSSEIN BATENI[†], MOHAMMAD TAGHI HAJIAGHAYI[‡], AND VAHID LIAGHAT[§]

Abstract. Moss and Rabani 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 node-weighted prize-collecting 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, node-weighted prize-collecting 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 leads to a much simpler algorithm than that of Moss and Rabani for PCST. Our second main contribution is for the budgeted node-weighted Steiner tree problem, which is also an improvement to the work of Moss and Rabani. In the unrooted case, we improve upon an existing $O(\log^2 n)$-approximation by Guha et al., 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 Moss and Rabani to the bicriteria approximation ratio of $(1 + \epsilon, O(\log n)/\epsilon^2)$ for any positive (possibly subconstant) ϵ. That is, for any permissible budget violation $1 + \epsilon$, we present an algorithm achieving a trade off in the guarantee for the prize. Indeed, we show that this is almost tight for the natural linear-programming relaxation used by us as well as in the previous works.

Key words. approximation algorithms, network design, Steiner connectivity, budgeted Steiner tree, node-weighted connectivity

AMS subject classifications. 68W25, 05C85, 68R10

DOI. 10.1137/15M102695X

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 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 a Steiner tree,

*Received by the editors June 19, 2015; accepted for publication (in revised form) January 29, 2018; published electronically July 5, 2018. A preliminary version of this paper appeared in *International Colloquium on Automata, Languages and Programming (ICALP)*, Springer, Heidelberg, 2013, pp. 81–92.
http://www.siam.org/journals/sicomp/47-4/M102695.html
Funding: This work was 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). The third author's work was supported in part by a Google Ph.D. Fellowship Award during parts of this study.
[†]Google Research, New York, NY 10011 (bateni@google.com).
[‡]Computer Science Department, University of Maryland, College Park, MD 20742 (hajiagha@cs.umd.edu).
[§]Facebook, Menlo Park, CA 94025 (vliaghat@fb.com).

the traveling salesman problem, and their variants) have been the testbed for new ideas and have been instrumental in the 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* (NW) problem, the goal is to maximize the prize of H minus its cost.[1] We prove in section 4 that this natural problem does not admit any finite approximation algorithm. 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 the *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 α, $0 < \alpha < 1$, be a constant. The following statements are equivalent (both for edge-weighted and node-weighted variants):*

(i) *There is an α-approximation algorithm for the rooted k-MST problem.*
(ii) *There is an α-approximation algorithm for the unrooted k-MST problem.*
(iii) *There is an α-approximation algorithm for the k-Steiner tree problem.*

Proof. Here we present the equivalence of (ii) and (iii) (see section 5 for that of (i) and (ii)). We note that one way is clear by definition. To prove that (ii) implies (iii), 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 has 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 the 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. 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 Erlebach, Grant, and Kammer [5]. For the earlier work on an 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]The prize or cost of a subgraph is defined as the total prize or cost of its vertices, respectively.

1.1. Contributions and techniques.

Approximation algorithm for prize collecting Steiner forest (PCSF). Klein and Ravi [11] were the first to give an $O(\log h)$-approximation algorithm for the Steiner forest (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 Steiner tree (ST) problem. The ST problem is a special case of SF where all demands share an endpoint. In an independent work, Chekuri, Ene, and Vakilian [2] give an algorithm with an approximation ratio of $O(\log n)$ w.r.t. 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 linear program (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 an $O(\log n)$ 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 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 Könemann, Sadeghian, and Sanita [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, Sadeghian, and Sanita [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 ST 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 a decade no one noticed the flaw in the algorithm of Moss and Rabani [13].[3] Indeed the recently proposed algorithm by Könemann, Sadeghian, and Sanita [12] is even more sophisticated on both phases; their algorithm is not monotone anymore. For the growth phase, their algorithm only connects an active moat (i.e., one whose growth has not reached its penalty), to an inactive moat, only if the total dual of participating components is large enough. For the pruning phase, they use a potential function argument to choose the final subtree.

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

[2] Introduced by Agrawal, Klein, and Ravi [1] and Goemans and Williamson [8].

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

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 the optimal solution (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, Sadeghian, and Sanita [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 of 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 ϵ), 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 section 6. In particular, there are instances for which the fractional solution is 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 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.

1.2. Organization. Next in section 2 we briefly discuss the method of Moss and Rabani for deriving an algorithm for the budget problem from that for PCST. We then explain and analyze our algorithm for PCSF in section 2.2. Section 3 discusses our trimming procedure and how it leads to improved results for budgeted problems.

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

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

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

Finally, sections 4 to 6 contain minor results for hardness of NW and reductions between special cases of the quota problem.

2. The PCSF 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 *SF* 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 $\mathcal{S}_i$ 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):

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

[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.

This IP is indeed the standard formulation studied in the literature. Given a combinatorial 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 $\mathbf{PCSF}(X)$: we first assume $\mathcal{T} \subseteq X$ since the cost of a terminal is zero. For every vertex $v \in X$, we set $\mathbf{x}(v) = 1$. This implies $\sum_{v\in\delta(S)} \mathbf{x}(v) \geq 1$ for every set S that separates a connected component of $G[X]$, hence, satisfying the IP constraint for such S. Now consider a set $S \in \mathcal{S}_i$ where S does not separate any connected component. Therefore, there exists a connected component CC of $G[X]$ containing an endpoint of demand i such that $CC \cap X = \phi$. To satisfy the IP constraint for all such S, we set $\mathbf{z}(V \backslash CC) = 1$ for every connected component CC of $G[X]$. This would satisfy $\sum_{R|S\subseteq R\in\mathcal{S}_i} \mathbf{z}(R) \geq 1$.

One can further confirm that the cost of the IP solution $\langle \mathbf{x}, \mathbf{z}\rangle$ is exactly $\mathbf{PCSF}(X)$: we buy the same vertices and pay $\sum_{v\in V\backslash\mathcal{T}} c(v)\mathbf{x}(v)$. For the penalty cost, the IP solution pays $\sum_{S\in\mathcal{S}} \pi(S)\mathbf{z}(S) = \sum_i \sum_{S\in\mathcal{S}_i} \frac{\pi_i}{2}\mathbf{z}(S)$. Consider an unsatisfied demand i with s_i (resp., t_i) in a connected componnent $CC(s_i)$ (resp., $CC(t_i)$). There are exactly two sets $S \in \mathcal{S}_i$ with $\mathbf{z}(S) = 1$: $V\backslash CC(s_i)$ and $V\backslash CC(t_i)$. Therefore the overall penalty of the solution is exactly the total penalty of unsatisfied demands.

The converse of the above argument can be used to show that any feasible solution $\langle \mathbf{x}, \mathbf{z}\rangle$ of the IP corresponds to a solution $X \subseteq V$ for the PCSF problem with at most the same cost. One can obtain a relaxed LP from IP by relaxing the integral constraint $\mathbf{x}(v), \mathbf{z}(S) \in \{0, 1\}$ to $\mathbf{x}(v), \mathbf{z}(S) \in [0, 1]$. Let OPT denote the objective value of the optimal solution for the relaxed LP. The following is the dual program $\mathcal{D}$ corresponding to the LP:

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

In the case of the ST, the dual variables are defined w.r.t. a set S. However, in the ST, 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 the ST 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.[8] 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

[8]Our algorithm in the next section runs in iterations in which we reduce the weights of nodes purchased in previous iterations to zero. Therefore, we need a careful and rigid handling of zero-weight nodes.

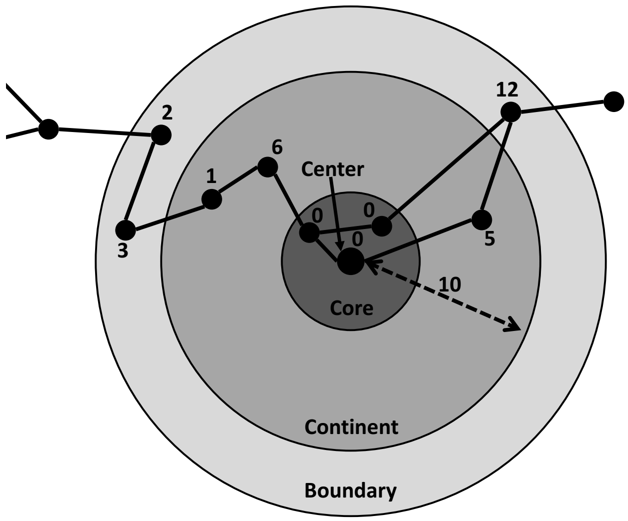

FIG. 1. *A graphical representation of a disk of radius* 10. *The vertex at the center of the disk is an endpoint of a demand. The numbers show the cost of vertices. The innermost circle contains the core, while the outermost circle contains the continent* and *the boundary.*

dual w.r.t. c and $\mathcal{L}$ is the following program $\overline{\mathcal{D}(c,\mathcal{L})}$:

$$(\overline{\mathcal{D}(c,\mathcal{L})}) \qquad \text{Maximize} \quad \sum_{S\in\mathcal{S}} \mathbf{y}(S),$$

$$\text{(C1)} \qquad \forall v \in V \quad \sum_{S\in\overline{\mathcal{S}(c,\mathcal{L})}:v\in\delta(S)} \mathbf{y}(S) \le c(v),$$

$$\text{(C2)} \qquad \forall S \in \overline{\mathcal{S}(c,\mathcal{L})} \quad \sum_{S':\mathbf{core}(S)\subseteq S'\subseteq S} \mathbf{y}(S') \le \pi_{\mathcal{L}}(S),$$

$$\mathbf{y}(S) \ge 0 \ .$$

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 necessarily inside the disk. See Figure 1 for a graphical representation of a 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 *nonoverlapping* if their union is a feasible dual solution (i.e., both sets of constraints C1 and C2 hold).

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

Proof. If a vertex v is inside a disk $\mathbf{y}_i$, the corresponding dual constraint C1 is tight for $\mathbf{y}_i$ (Fact 2). Thus for any set S such that $v \in \delta(S)$, the variable $\mathbf{y}_{-i}(S) := \sum_{j\in[k]\setminus\{i\}} \mathbf{y}_j(S)$ has to be zero, otherwise $\mathbf{y}$ won't be feasible. Hence, v cannot be in the continent of another disk. On the other hand, since the distance between v and the center is *strictly* less than the radius, there exists a set S^* containing v with positive dual value in $\mathbf{y}_i$. By Fact 3, S^* is a subset of the continent of $\mathbf{y}_i$.

Now by contradiction, assume that v is on the boundary of another disk $\mathbf{y}_j$. Let c_j and r_j denote the center and radius of $\mathbf{y}_j$, respectively. By definition, v has a neighbor u such that $d(c_j, u) \leq r_j$. The dual constraint C1 for u is already tight in $\mathbf{y}_j$. Hence, the same argument as above holds for u as well: u cannot be in the continent of $\mathbf{y}_i$. Therefore, u lies in $\delta(S^*)$. Since $\mathbf{y}_i(S^*)$ is positive, the dual vector $\mathbf{y}_i + \mathbf{y}_j$ cannot be feasible, which is a contradiction. ◻

Proposition 1 implies that in the union of a set of nonoverlapping disks, the continents are pairwise far from each other. This intuition leads to the following.

LEMMA 1. *Suppose T' is a maximal subset of terminals such that the distance between every pair of them is nonzero. Let R denote the maximum radius such that the $|T'|$ disks of radius R centered at terminals in T' are nonoverlapping. 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.*

Proof. Let $\mathbf{y}_1, \ldots, \mathbf{y}_{|T'|}$ denote the disks of radius R centered at the terminals in T' with cores $core_1, \ldots, core_{|T'|}$. Increasing the radius of all disks by any $\epsilon > 0$ creates an infeasibility in their union $\mathbf{y}$. Thus at least one of the following holds for $\mathbf{y}$:

- *The constraint* C2 *is tight for a set S containing a terminal, i.e., $\sum_{S'\subseteq S} \mathbf{y}(S') = \pi(S) > 0$.* Let S be such a set with the smallest cardinality. Recall that by Fact 3, $\mathbf{y}_i(S')$ is positive for a set S' only if S' contains the center of $\mathbf{y}_i$ and is a subset of the continent of the disk. We remove the zero terms from both sides of the equality. The right-hand side would be the penalty of a subset of the terminals, say T''. The left-hand side would be the sum over dual variables $\mathbf{y}(S')$'s such that S' is a subset of a continent of one of the disks centered on $T'' \cap T'$. For each disk centered on a terminal i, we are guaranteed that

$$\sum_{S\subseteq continent\ of\ the\ disk} \mathbf{y}(S) = \sum_{S\subseteq continent\ of\ the\ disk} \mathbf{y}_i(S) <= \pi(core_i).$$

Therefore if the inequality is tight overall, it has to be tight induced to any disk in T''. Thus the smallest set S is indeed a subset of the continent of a single disk. Now let S^* be the continent of that disk. The sets S and S^* share the same core, thus the right-hand sides of the constraint C2 for both are the same. However the left-hand side of the constraint for S^* can only be larger which leads to (i).

- *For a vertex v the constraint* C1 *becomes infeasible if we grow every disk by any $\epsilon > 0$.* The constraint for v is tight w.r.t. $\mathbf{y}$. If the constraint for v is not tight in any of the $\mathbf{y}_i$'s independently, then v is on the boundary of more than one disk which leads to (ii). Otherwise assume the constraint for v is tight in $\mathbf{y}_i$ for an $i \in [|T'|]$.[9] If we extend the radius by ϵ, v will be inside the ith disk thus $\sum_{S|v\in\delta(S)} \mathbf{y}_i(S)$ will not change. However by the assumption about v, the same summation for $\mathbf{y}$, i.e., $\sum_{S|v\in\delta(S)} \mathbf{y}(S)$ will increase. Therefore a neighbor of v is at most R far from the center of another disk, say that of the jth disk. By definition, v is on the boundary of the jth disk. Further, by Proposition 1, v cannot be inside the ith disk and so is on its boundary which leads to (ii). ∎

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. The proof is based on the analysis of the growth of a disk.

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 $\sum_{S|v\in\delta(S)} \mathbf{y}(S) = R - (d^c(t,v) - c(v))$.*

Proof. Consider the continuous process of growing the disk during which we start with a set S^* (initialized to the core containing t), and we add the vertices to S^* for which the constraint C1 becomes tight. Let $\langle v_0 = t, v_1, \dots, v_m\rangle$ denote the set of vertices in the continent or the boundary of the disk, sorted by their distance to t. We prove by induction that for every i, (a) $\sum_{S|v_i\in\delta(S)} \mathbf{y}(S) = \min\{d(t,v_i), R\} - (d(t,v_i) - c(v_i))$ and (b) v_i enters the growing set S^* when the total growth of the disk has reached $d(t,v_i)$. Note that given (a) for all vertices, (b) follows.

The base is trivial. Consider an arbitrary $i > 0$. Let v_j be the closest neighbor of v_i to the center t, thus $\min\{d(t,v_i), R\} - (d(t,v_i) - c(v_i)) = \min\{d(t,v_i), R\} - d(t,v_j)$ and $j < i$. The vertex v_i falls on $\delta(S^*)$ when v_j enters the growing set S^*. This happens when the constraint for v_j is tight. The induction hypothesis implies that the total growth of the disk is $d(t,v_j)$. Any further growth of the disk contributes to $\sum_{S|v_i\in\delta(S)} \mathbf{y}(S)$, until either of the following happen:

- The total growth reaches R. Thus

$$\sum_{S|v_i\in\delta(S)} \mathbf{y}(S) = R - d(t,v_j) = R - (d(t,v_i) - c(v_i)).$$

- The constraint for v_i becomes tight. Thus

$$\sum_{S|v_i\in\delta(S)} \mathbf{y}(S) = c(v_i) = d(t,v_i) - d(t,v_j) = \min\{d(t,v_i), R\} - d(t,v_j),$$

which completes the proof. ∎

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 $\mathcal{D}_i = \mathcal{D}(c_i, \mathcal{L}_i)$. The

[9] For an integer x, let $[x]$ denote the set $\{1, 2, \dots, x\}$.

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 more restrictive than $\mathcal{D}$: $\mathcal{D}_i$ has the same set of constraints, but the bounds on the packing constraints can only be smaller. 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 nonoverlapping 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 τ (and so to the core containing τ). 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.

Algorithm 1 The PCSF algorithm.

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

Initialize $X_0 = \mathcal{T}$, $\mathcal{L}_1 = \mathcal{L}$, $c_1 = c$, and $i = 1$.
while $|\mathcal{L}_i| > 0$ **do**
  Set $c_i = c[X_{i-1} \to 0]$ and construct the dual program $\mathcal{D}_i$ with respect to c_i and $\mathcal{L}_i$.
  Construct T_i by choosing an arbitrary terminal from each core.
  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. $\mathcal{D}_i$.
  if the constraint C2 is tight for a continent S **then**
    Set $X_i = X_{i-1}$.
    Set $\mathcal{L}_{i+1} = \mathcal{L}_i \setminus \{j \in [h] |$ *either* $s_j \in \mathbf{core}(S)$ *or* $t_j \in \mathbf{core}(S)\}$.
  else
    Find a vertex v on the boundary of multiple disks for which constraint C1 is tight.
    Let L_v denote the centers of the disks whose boundaries contain v.
    Initialize $X_i = X_{i-1}$.
    for all $\tau \in L_v$ **do**
      Add the shortest path (w.r.t. c_i) between τ and v to X_i.
    Set $\mathcal{L}_{i+1} = \mathcal{L}_i \setminus \{j \in [h] | d^{c_{i+1}}(s_j, t_j) = 0\}$.
  $i = i + 1$.
Output X_{i-1}.

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.

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 1. *$P_i \leq 2h_iR_i$ 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 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 $h_i \geq |L_v| - 1$.[10] Recall that $|L_v| \geq 2$ 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 τ 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:

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

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

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

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

[10]In the special case that every endpoint in the cores become connected to the other endpoint of its demand, $h_i = |L_v|$; otherwise $h_i = |L_v| - 1$.

3. The budgeted ST problem. In this section we consider the budgeted problem in the node-weighted ST 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.

LEMMA 3. *Let T be a subtree rooted at r with the prize-to-cost ratio γ. 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 $\frac{\epsilon}{2}B \leq c(T^*) \leq (1+\epsilon)B$.*

Proof. Consider T rooted at r. As an initial step, we repeatedly remove a subtree of T if (i) the (prize-to-cost) ratio of the remaining tree is at least γ, and (ii) the cost of the remaining tree is at least $\frac{\epsilon B}{2}$. We repeat this until no such subtree can be found.

If the current cost of T is at most $(1+\epsilon)B$ we are done. Suppose it is not the case. A subtree T' is *rich* if $c(T') \geq \frac{\epsilon}{2}B$ and the ratio of T' and all its subtrees is at least γ. Indeed the existence of a rich subtree proves the lemma.

CLAIM 2. *Given a rich subtree T', the desired tree T^* can be found.*

Proof. Find a rich subtree $T'' \subseteq T'$ such that the strict subtrees of T'' are not rich, i.e., $c(T'') \geq \frac{\epsilon}{2}B$ while the cost of strict subtrees of T'' (if any exist) is less than $\frac{\epsilon}{2}B$. Let C denote the total cost of the immediate subtrees of T''. We distinguish between two cases.

- If $C < \frac{\epsilon}{2}B$, then we can connect the root of T'' directly to r. The cost of the resulting tree is at most $C + B \leq (1+\epsilon)B$. On the other hand, T'' is rich thus the prize of T'' is at least $\gamma\left(\frac{\epsilon}{2}B\right)$. Therefore the resulting tree has the desired ratio $\frac{\gamma\epsilon}{2(1+\epsilon)} \geq \frac{\gamma\epsilon}{4}$.
- If $C \geq \frac{\epsilon}{2}B$, we can pick a subset of immediate subtrees of T'' such that their total cost is between $\frac{\epsilon}{2}B$ and ϵB. We connect these subtrees to the root by picking the path from the root of T'' to r. Using the same argument as above, one can show that the resulting tree has the desired properties. ∎

It only remains to consider the case that no rich subtree exists. Since T is not rich, the ratio of at least one subtree is less than γ. Find a subtree T' such that the ratio of T' is less than γ while the ratio of all of its strict subtrees (if any exist) is at least γ. Though the ratio of T' is low, we have not removed it in the initial step. Thus the cost of $T \backslash T'$ is less than $\frac{\epsilon}{2}B$. However, $c(T) > (1+\epsilon)B$ and thus the total cost of immediate subtrees of T' is at least $\frac{\epsilon}{2}B$. On the other hand the cost of an immediate subtree of T' is less than $\frac{\epsilon}{2}B$, otherwise it would be a rich subtree. Therefore we can pick a subset of immediate subtrees of T' such that their total cost is between $\frac{\epsilon}{2}B$ and ϵB. We connect these subtrees by connecting the root of T' directly to r. The resulting tree has the cost at most $(1+\epsilon)B$ and the prize at least $\gamma\left(\frac{\epsilon}{2}B\right)$ which completes the proof. ∎

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 .

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 an $\Omega(\frac{\epsilon^2}{\log n})$ fraction of OPT.*

Proof. First we make the graph B proper for the root r by simply discarding the vertices which are farther than B from r. Note that these vertices cannot be a part of an optimal solution. Now we first use the following theorem proved by Moss and Rabani [13].

Theorem 12 of [13]. *For an instance of the rooted budgeted problem, an $O(\log n)$-approximation solution can be found in polynomial time which uses at most twice the budget.*

By the above theorem, we can find a tree T' with $\pi(T') \geq \frac{\text{OPT}}{O(\log n)}$ and $c(T') \leq 2B$. Suppose the cost of T' is more than $(1+\epsilon)B$; otherwise we are done. Let $\gamma(T')$ denote the prize-to-cost ratio of T'. Observe that $\gamma(T') = \frac{\pi(T')}{c(T')} \geq \frac{\text{OPT}}{O(\log n)\cdot 2B}$. By Lemma 3 we can trim T' to obtain a subtree T such that

- the prize-to-cost ratio of T is $\gamma(T) \geq \frac{\epsilon}{4}\gamma(T') \geq \frac{\epsilon \text{OPT}}{O(\log n)B}$;
- the cost of T is sandwiched between $\frac{\epsilon}{2}B$ and $(1+\epsilon)B$.

Therefore the cost of T does not violate the budget by much and $\pi(T)$ is at least

$$\pi(T) \geq \gamma(T)\left(\frac{\epsilon}{2}B\right) \geq \frac{\epsilon^2\text{OPT}}{O(\log n)} .$$

∎

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. For the sake of completeness, we present the proof here in detail.

LEMMA 4. *Let T be a tree with the prize-to-cost ratio γ. 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 $\frac{B}{4} \leq c(T^*) \leq B$.*

Proof. We make T rooted at an arbitrary vertex r. As the first pruning step, we repeatedly discard a subtree if the ratio and the cost of the remaining tree does not go below γ and $\frac{B}{4}$, respectively. We stop when no such subtree can be found. Suppose the current cost of T is more than B; otherwise we are done. As in Lemma 3, a subtree T' is *rich* if the ratio of T' and all subtrees of T' is at least γ. Note that one can easily check whether a subtree is rich.

First we show, given a rich subtree, we can easily find the solution. Observe that all the subtrees of a rich subtree are also rich unless their cost is less than $\frac{B}{4}$. Given a rich subtree, let T' be its lowest rich subtree, i.e., the cost of any immediate subtree of T' (if any exists) is less than $\frac{B}{4}$. Now let C denote the total cost of all immediate subtrees of T'.

- If $C < \frac{B}{4}$ (or no child exists), then $c(T') \le \frac{3B}{4}$ since the cost of the root of T' does not exceed $\frac{B}{2}$. Thus T' satisfies the properties desired in the lemma. Recall that T' is rich and thus its ratio is at least γ.
- If $C \ge \frac{B}{4}$, we can pick a subset of immediate subtrees of T' such that their total cost is between $\frac{B}{4}$ and $\frac{B}{2}$. This can be done since the cost of an immediate subtree is at most $\frac{B}{4}$. Let T^* be the tree formed by connecting these subtrees to the root of T'. Observe that $c(T^*) \le B$ and the total prize is at least $\pi(T^*) \ge \gamma\frac{B}{4}$. Therefore, the ratio of T^* is at least $\frac{\gamma}{4}$.

It only remains to consider the case that T does not have a rich subtree. Since T is not rich, a subtree of T has ratio less than γ. Let T' be a subtree with ratio less than γ such that all strict subtrees of T' (if any exist) have ratio at least γ. Observe that the cost of an immediate subtree of T' is less than $\frac{B}{4}$, otherwise it would be a rich subtree. On the other hand, we have not discarded T' in the first pruning step, hence, $c(T \backslash T') < \frac{B}{4}$. Furthermore $c(T) > B$, thus the total cost of immediate subtrees of T' is at least $\frac{B}{4}$. Now similarly to the previous argument, we can pick a subset of immediate subtrees of T' such that their total cost is between $\frac{B}{4}$ and $\frac{B}{2}$. The tree formed by connecting these subtrees to the root of T' has the desired properties. ◻

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 indeed shows 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 within a constant factor of OPT. 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 3. *Either $\pi(T_f^*) \ge \frac{OPT}{2}$ or $\pi(T_s^*) \ge \frac{OPT}{2}$.*

Proof. Let T^* denote the optimal tree. If T^* consists of only one vertex, then clearly it is either a flat tree or a saddled tree and we are done. Now assume that T^* is neither flat nor saddled and it has at least two vertices. Let x and y denote the vertices with the maximum cost and the second maximum cost in T^*, respectively. Since T^* is not flat we have $c(x) > \frac{B}{2}$ and $c(y) \le \frac{B}{2}$. Neither are saddled, thus $c(y) > \frac{B-c(x)}{2}$. Observe that the cost of any other vertex of T^* is at most $\frac{B-c(x)}{2}$. Consider the path between y and x in T^*. Let e denote the edge of the path which is adjacent to y. Removing e from T^* results in the two subtrees T_y and T_x containing y and x, respectively. The cost of every vertex in T_y is at most $c(y) \leq \frac{B}{2}$, thus T_y is flat. On the other hand the cost every vertex in T_x except x is at most $\frac{B-c(x)}{2}$, thus T_x is saddled. This completes the proof since one of the subtrees has at least half the optimal prize $\pi(T^*)$. ∎

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][11] 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. ∎

4. Hardness of NW. Here we present the hardness result for the rooted NW and directed NW problem given in Feigenbaum, Papadimitriou, and Shenker [6] with slight modifications. We show that the NW is NP-hard to approximate within any finite factor when restricted to the case of bounded degree graphs.

THEOREM 5. *For any ϵ, $0 < \epsilon < 1$, it is NP-hard to approximate*[12] *the rooted, whether directed or undirected, NW problem within a ratio ϵ.*

Proof. Given an instance I of 3-SAT, we make an instance J of an NW problem such that (i) if I is a yes-instance (i.e., it is satisfiable), then an ϵ-approximation answer to J is strictly greater than ϵ; and (ii) if I is a no-instance, then the optimal answer to J is at most ϵ. Let n and m be the number of variables and clauses in I, respectively. Without loss of generality we assume that for every variable x there is a clause $x \vee \bar{x}$ in I, thus $m \geq n+1$. We make the instance J with four layers of vertices as follows:

- In the top layer, we put the root r with prize $\pi(r) = \epsilon$.
- In the next layer, we put a vertex r' with prize zero, connected to r via an edge of cost $mK - (n+1) - m$ for a fixed $K \geq n+1$.
- The third layer contains $2n$ vertices, for every literal in I, all with prize zero and connected to r' via edges of unit cost.
- The last layer contains m vertices, one for every clause, all with prize K and connected to the vertices corresponding to the literals it contains via edges of unit cost.

In the case of directed NW, we orient all the edges from top to bottom.

We claim that if I is satisfiable, then $NW(J) \geq 1 + \epsilon$, otherwise $NW(J) \leq \epsilon$. Note that in the former an ϵ-approximation algorithm would give us a solution with NW at least $\epsilon(1+\epsilon) > \epsilon$ and in the latter it would give us a solution with NW at most ϵ, thus it can distinguish the satisfiability of I.

[11]See the statement of their theorem in the proof of Theorem 3.

[12]Algorithm A approximates function f within a ratio ϵ iff for every input instance x, $\epsilon f(x) \leq A(x) \leq f(x)/\epsilon$. Since NW is a maximization problem, we can assume that $A(x) \leq f(x)$.

First suppose I is satisfiable. Connect r to r' and r' to all literals satisfied in the solution. Finally connect each clause to one of its satisfied literals. The total prize in this solution is $mK + \epsilon$, and the total cost is $mK - (n+1) - m + n + m = mK - 1$. Therfore the NW is $1 + \epsilon$, as desired.

Next we demonstrate that NW larger than ϵ implies a satisfying assignment for I. Clearly such a solution buys the edge (r, r') and thus incurs a big cost. Suppose this solution includes vertices S in the third level and vertices T in the last level. The total prize collected is $\epsilon + |T|K$. We now give a lower bound on the cost of the solution.

Note that the edge weights in the subgraph under r' are all one. Any connected subgraph with $|S| + |T| + 1$ vertices (including r' itself) costs at least $|S| + |T|$. The maximum NW we could get is

$$\begin{aligned}&\epsilon + |T|K - |S| - |T| - (mK - (n+1) - m)\\&\quad = \epsilon + |T|(K-1) - m(K-1) + (n + 1 - |S|)\\&\quad = \epsilon + (|T| - m)(K-1) + (n + 1 - |S|),\end{aligned}$$

which can be larger than ϵ only if $|T| \geq m$, since $|S| \geq 1$ and $K \geq n + 1$.

Therefore, to have a net worth strictly more than ϵ, we need to include all the vertices in the fourth layer. On the other hand, observe that even with $|T| = m$, we need $|S| \leq n$ to allow for net worth strictly greater than ϵ.

Recall that for every variable x there is a clause $x \vee \bar{x}$. To include the vertex corresponding to this clause, we need to include at least one vertex corresponding to a literal of x. Coupled with the above observation, this shows that for every variable x, the vertex r' would be connected to exactly one of the vertices corresponding to x and $\bar{x}$.

Therefore a solution of NW more than ϵ corresponds to a satisfying assignment, as desired. In fact, the NW of such a solution would be exactly $1 + \epsilon$. □

5. Reductions for quota problems. In this section, we present two important reductions that were missing in the literature. More specifically, we show that rooted and unrooted k-MST and their k-ST versions are all equivalent and indeed equivalent to the quota problem. (That these are simpler than the latter is easy.) These results improve the approximation ratio of k-ST from 4 to 2. Ravi et al. [14] had provided a reduction from k-ST to k-MST losing a factor of 2, whereas Chudak, Roughgarden, and Williamson [4] had conjectured the presence of a $(2+\epsilon)$-approximation algorithm while presenting one with an approximation ratio of 5.

This section deals with four special cases of the quota node-weighted ST problem. We first claim that the rooted k-ST problem is equivalent to the quota problem, with a factor of $1 + \epsilon$ for a polynomially small ϵ. That the former is a special case of the latter can be observed easily by setting vertex prizes to 0 and 1 for Steiner and terminal nodes, respectively, and looking for a prize of at least k. To establish the other direction of the reduction, given a graph G with prize π and cost c on its vertices, as well as target prize value P, we produce an instance of k-ST as follows. We assume all vertices of G are Steiner vertices and connect a vertex u to $q(u) = \lceil \frac{n\pi(u)}{\epsilon P} \rceil$ new terminal vertices of cost zero. In this instance we let $k = \lfloor n/\epsilon \rfloor$. Clearly any solution to the quota instance turns into a solution of k-ST if one collects the terminals immediately connected to the solution vertices. Next consider a solution to the k-ST instance. We can assume without loss of generality that either none or all of the terminals connected to one node are in the solution. The solution to the quota instance simply includes all Steiner nodes whose all adjacent terminals are picked in the k-ST instance. For such nodes we have $\sum_u q(u) \geq k$. Note that there are at most n such

Steiner nodes, and for each of them, say u, we have $q(u) \cdot \frac{\epsilon P}{n} < \frac{\epsilon P}{n} + \pi(u)$. Therefore, we get $(\frac{\epsilon P}{n}) \sum_u q(u) < \epsilon P + \sum_u \pi(u)$. However, the solution guarantee (in the k-MST instance) is that the left-hand side is at least $(\frac{\epsilon P}{n})k > (\frac{\epsilon P}{n})(\frac{n}{\epsilon} - 1) = P - \frac{\epsilon P}{n}$. Putting these two together and noting that $n \geq 1$, we obtain $\sum_u \pi(u) > P(1 - 2\epsilon)$.

The following two theorems show the other three problems are equivalent to k-ST (and hence the quota problem).

THEOREM 6. *Let α, $0 < \alpha < 1$, be a constant. The following two statements are equivalent:*
(i) *There is an α-approximation algorithm for the rooted k-*MST *problem.*
(ii) *There is an α-approximation algorithm for the unrooted k-*MST *problem.*

Proof. We note that by running the rooted k-MST for every vertex, (i) immediately implies (ii). To prove that (ii) implies (i), we give a cost-preserving reduction from rooted variant to unrooted variant. Let $< G = (V, E), r, k >$ be an instance of the rooted k-MST and let $n = |V|$. We add n vertices to G, all connected by edges of cost zero to r. Let $k' = k + n$ and consider the solution to (unrooted) k'-MST on the new graph. Since $k' > n - 1$, a subtree of size k' has to include r. Thus we can assume that there exist an optimal solution which includes all the n extra vertices plus a minimum-cost subtree of size k rooted at r. Hence the reduction preserves the cost of the optimal solution. ∎

THEOREM 7. *Let α, $0 < \alpha < 1$, be a constant. The following two statements are equivalent:*
(i) *There is an α-approximation algorithm for the k-ST problem.*
(ii) *There is an α-approximation algorithm for the k-*MST *problem.*

Proof. We note that one way is clear by definition. To prove that (ii) implies (i), similarly to Theorem 6, we give a cost-preserving reduction from k-ST to k-MST. Let $< G = (V, E), T, k >$ be an instance of k-ST 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 has 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 the optimal solution. ∎

All the above proofs work, mutatis mutandis, for the edge-weighted case, too.

6. Integrality gap for budgeted ST. In this section we discuss the linear programming approach to the budgeted problem. Let P_v denote the set of all paths from root to vertex v. We may also assume that all the edges have unit length. Consider the flow-based linear programming below:

$$\text{Maximize} \quad \sum_{v \in V} \pi_v \sum_{p \in P_v} \mathbf{f}_p,$$

$$\text{(X)} \qquad \forall e \in E, v \in V \quad \sum_{p \in P_v : e \in p} \mathbf{f}_p \leq \mathbf{x}_e,$$

$$\text{(F)} \qquad \forall v \in V \quad \sum_{p \in P_v} \mathbf{f}_p \leq 1,$$

$$\text{(B)} \qquad \sum_{e \in E} \mathbf{x}_e \leq B,$$

$$\mathbf{f}_p, \mathbf{x}_e \geq 0.$$

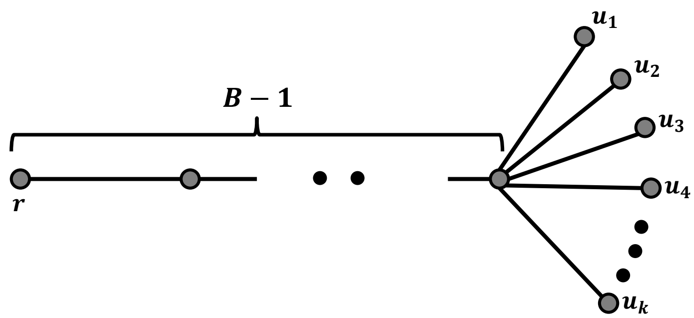

FIG. 2. *An example showing the unbounded gap of the LP for the budget problem.*

Intuitively, for a path p ending at v, $\mathbf{f}_p$ denotes the total flow reaching v through p and $\mathbf{x}_e$ denotes the maximum flow passing through the edge e. Constraint B keeps the cost of edges in budget and constraint F restricts the total flow reaching a vertex. One can also write a similar cut-based LP. However, we can show that even if G is a tree, the gap between the fractional and integral solutions is unbounded. Let G be a tree obtained by putting a star at the end of a long path of length $B-1$ (see Figure 2). Let $u_1, \ldots, u_k$ denote the leaves other than the root which have 1 unit of profit. Other vertices have zero profit. Clearly the optimal integral solution gains one unit of profit. Let p_i denote the path from r to u_i. Consider a feasible fractional solution where for every i, $f_{p_i} = \frac{B}{B+k-1}$ and, therefore, for every edge e, $x_e = \frac{B}{B+k-1}$. We note that since there are $B+k-1$ edges, we are not exceeding the budget. This shows that the optimal fractional solution is at least $\frac{kB}{B+k-1}$ and, hence, in case of $B \geq k$, the gap between the fractional and the integral solution is k.

REFERENCES

[1] A. AGRAWAL, P. KLEIN, AND R. RAVI, *When trees collide: An approximation algorithm for the generalized Steiner problem on networks*, in STOC, ACM, New York, 1991, pp. 134–144.

[2] C. CHEKURI, A. ENE, AND A. VAKILIAN, *Prize-collecting survivable network design in node-weighted graphs*, in APPROX, Springer, Berlin, 2012, pp. 98–109.

[3] X. CHENG, Y. LI, D.-Z. DU, AND H. Q. NGO, *Steiner trees in industry*, in Handbook of Combinatorial Optimization, Springer, New York, 2005, pp. 193–216.

[4] F. A. CHUDAK, T. ROUGHGARDEN, AND D. P. WILLIAMSON, *Approximate k-MSTs and k-Steiner trees via the primal-dual method and Lagrangean relaxation*, Math. Program., 100 (2004), pp. 411–421, https://doi.org/10.1007/s10107-003-0479-2.

[5] T. ERLEBACH, T. GRANT, AND F. KAMMER, *Maximising lifetime for fault-tolerant target coverage in sensor networks*, in SPAA, ACM, New York, 2011, pp. 187–196.

[6] J. FEIGENBAUM, C. H. PAPADIMITRIOU, AND S. SHENKER, *Sharing the cost of multicast transmissions*, J. Comput. System Sci., 63 (2001), pp. 21–41.

[7] N. GARG, *Saving an epsilon: A* 2*-approximation for the k-MST problem in graphs*, in STOC, ACM, New York, 2005, pp. 396–402.

[8] M. X. GOEMANS AND D. P. WILLIAMSON, *A general approximation technique for constrained forest problems*, SIAM J. Comput., 24 (1995), pp. 296–317.

[9] S. GUHA, A. MOSS, J. S. NAOR, AND B. SCHIEBER, *Efficient recovery from power outage*, in STOC, ACM, New York, 1999, pp. 574–582.

[10] K. JAIN AND V. V. VAZIRANI, *Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and Lagrangian relaxation*, J. ACM, 48 (2001), pp. 274–296.

[11] P. KLEIN AND R. RAVI, *A nearly best-possible approximation algorithm for node-weighted Steiner trees*, J. Algorithms, 19 (1995), pp. 104–115.

[12] J. Könemann, S. Sadeghian, and L. Sanita, *An LMP $O(\log n)$-approximation algorithm for node weighted prize collecting Steiner tree*, in FOCS, IEEE, Piscataway, NJ, 2013, pp. 568–577.

[13] A. Moss and Y. Rabani, *Approximation algorithms for constrained node weighted Steiner tree problems*, SIAM J. Comput., 37 (2007), pp. 460–481.

[14] R. Ravi, R. Sundaram, M. V. Marathe, D. J. Rosenkrantz, and S. S. Ravi, *Spanning trees–short or small*, SIAM J. Discrete Math., 9 (1996), pp. 178–200.