Approximation Algorithms for Constrained Node Weighted Steiner Tree Problems

A. MOSS AND Y. RABANI

Abstract

We consider a class of optimization problems, where the input is an undirected graph with two weight functions defined for each node, namely the node's *profit* and its *cost*. The goal is to find a connected set of nodes of low cost and high profit. We present approximation algorithms for three natural optimization criteria that arise in this context, all of which are NP-hard. The *budget* problem asks for maximizing the profit of the set subject to a budget constraint on its cost. The *quota* problem requires minimizing the cost of the set subject to a quota constraint on its profit. Finally, the *prize collecting* problem calls for minimizing the cost of the set plus the profit (here interpreted as a penalty) of the complement set. For all three problems, our algorithms give an approximation guarantee of $O(\log n)$, where n is the number of nodes. To the best of our knowledge, these are the first approximation results for the quota problem and for the prize collecting problem, both of which are at least as hard to approximate as set cover. For the budget problem, our results improve on a previous $O(\log^2 n)$ result of Guha, Moss, Naor, and Schieber. Our methods involve new theorems relating tree packings to (node) cut conditions. We also show similar theorems (with better bounds) using edge cut conditions. These imply bounds for the analogous budget and quota problems with edge costs which are comparable to known (constant factor) bounds.

1 Introduction

We consider optimization problems on graphs with node costs and profits. Let $G = (V, E)$ be an undirected graph. Let every node $v \in V$ have a non-negative cost $c(v)$ and a non-negative profit $\pi(v)$. Our goal is to find a connected set of nodes $S \subset V$ that has low cost and high profit. There are several ways to cast this goal as an optimization criterion. In this paper we consider three such problems. In the *quota* problem, we are given a profit quota Q, and we have to find a set S of minimum total cost among those with total profit at least Q. In the *budget* problem, the total cost is constrained to be at most a budget B. Subject to this constraint, we have to find a set S of maximum total profit. Finally, in the *prize collecting* problem, the objective is to minimize the total cost of S plus the total profit of $V \setminus S$ (the profit loss). All three problems also have a *rooted* version, where S must contain a specified node $r \in V$. Clearly, an algorithm for the rooted version can be used to solve the unrooted version, by enumerating over all possible roots.

We give $O(\log |V|)$ approximation guarantees for the rooted (and thus the unrooted) quota problem, for the (rooted) budget problem, and for the (rooted) prize collecting problem. Our result for the budget problem is a bicriteria approximation in the sense that the approximation algorithm is allowed to exceed the budget by a factor of 2. The quota problem, as well as the prize collecting problem, are at least as hard to approximate as set cover, so our guarantees are asymptotically the best possible under reasonable assumptions on the complexity of NP-hard problems (see, for example, the survey by Arora and Lund(1997)). By the same token, the budget problem is at least as hard to approximate as maximum coverage, so there is a constant lower bound on its approximability under the same assumptions (see Khuller, Moss, and Naor (1998)). In addition to these new results, our methods can be used to derive previously discovered approximation guarantees for some related problems. Our solutions are more uniform, and perhaps more elegant, than previous results. See the discussion below for more details.

Most previous related research was done on closely related problems with edge costs instead of node costs. (The cost of a set of nodes is the total cost of its minimum spanning tree.) These problems are NP-hard, and in recent years constant factor approximation algorithms were found for most of them. Goemans and Williamson (1995) give a primal-dual $2 - \frac{1}{n-1}$ approximation algorithm for the prize collecting Steiner tree problem, which is the edges costs version of our prize collecting problem. Following the ground-breaking work of Blum, Ravi, and Vempala (1996), Garg (1996) gives a 3-approximation (as well as a simpler 5-approximation) for k-MST, which is a special case of the edge costs version of our quota problem, with unit profits and a quota of k. Arora and Karakostas (2000) improve Garg's approximation guarantee to $2 + \epsilon$, for all $\epsilon > 0$. The constant factor approximations of the latter three works are derived by using the Goemans-Williamson prize collecting Steiner tree algorithm on the graph with uniform profits. The profit is determined (using binary search) to yield a tree (more precisely, a convex combination of two trees) of size k. As noted by Chudak, Roughgarden, and Williamson (2001), Garg's k-MST 5-approximation algorithm can be recast in terms of a Lagrangian relaxation for the problem, similar to the treatment by Jain and Vazirani (1999) of the k-median problem.

Awerbuch, Azar, Blum, and Vempala (1995) (in a previous k-MST paper giving polylogarithmic approximation guarantees) note that a polynomial time k-MST approximation algorithm can be used to get a pseudo-polynomial time approximation algorithm with the same guarantee for the (edge costs) quota problem. Johnson, Minkoff, and Phillips (2000) note that for the above-mentioned constant approximation k-MST algorithms, the pseudo-polynomial time algorithm can be converted into a truly polynomial time one. Moreover, they show that such an algorithm can be used to solve the (edge costs) unrooted budget problem, losing a constant factor in the approximation guarantee.

The budget problem (with node costs) was introduced by Guha, Moss, Naor, and Schieber (1999), who motivate it by applications to problems in maintenance of electric power networks. They give an $O(\log^2 |V|)$ approximation guarantee for the unrooted version. They also show how to obtain a similar approximation guarantee for the rooted version using a set that has total cost at most twice the allowed budget B. Their algorithm is based on an $O(\log |V|)$ approximation algorithm for the node weighted Steiner tree problem by Klein and Ravi (1995). This problem is similar to the well-known Steiner tree problem, except that nodes and not edges have costs. The problem is at least as hard to approximate as set cover. As pointed out by Guha et al., the algorithm of Klein and Ravi is implicitly a primal-dual algorithm.

The starting point for all our algorithms is a standard linear programming relaxation for the rooted version, where nodes are chosen fractionally to be in the output set S, and the connectivity requirement is expressed as constraints on (node) cuts. For the quota and budget problems we show that any feasible solution to the linear programming relaxation can be approximated by a convex combination of connected sets of nodes containing the root. Moreover, such a convex combination is computable in polynomial time. It follows from an averaging argument that a good set can be found. Our algorithm for the prize collecting problem is used to construct the convex combination. It combines ideas from both the Klein-Ravi node weighted Steiner tree algorithm (with its primal-dual interpretation by Guha et al.) and the Goemans-Williamson prize collecting Steiner tree algorithm. Both the region growing process and the final delete step are more complicated in our algorithm than in either of those algorithms.

We then use the solution to the prize collecting problem to prove the following theorem:

Theorem 1. Let $G = (V, E)$ be an undirected graph with non-negative node weights $d : V \to \mathbb{Q}^+$. Let $r \in V$, and assume that for every $v \in V$, $d(v) \leq 1$ and $d(r) = 1$. Furthermore assume that for every node $v \in V \setminus \{r\}$, the minimum weight node cut separating r and v has weight of at least $d(v)$. Then, there exist polynomial time algorithms that compute

1. a packing in G of connected sets containing r such that for every node $v \in V$, the total weight of sets containing v is between $d(v)/c \log |V|$ and $d(v)$; and,

2. a packing in G of connected sets containing r such that for every node $v \in V$, the total weight of sets containing v is between $d(v)$ and $\min\{1, d(v)c \log |V|\}$.

where, in both cases, c is an absolute constant. Notice that the second claim implies the first. We state both claims for ease of application.

For optimization problems that can be formulated as positive integer programs, Carr and Vempala (2000) prove that an approximation algorithm for the problem can be used to generate a packing of integer solutions that approximates a feasible solution to the LP relaxation of the integer program with the same guarantee. Our results extend their results, and use an approximation algorithm for a different problem to generate the packing. Our approximation guarantees for the budget and quota problems follow from applying the packing algorithms to the solutions for the linear programming relaxations we use, and then using averaging arguments to find good integer solutions.

We can also show an analogous packing theorem for the case of edge capacities.

Theorem 2. Let $G = (V, E)$ be an undirected graph with non-negative node weights $d : V \to \mathbb{Q}^+$ and non-negative edge capacities $c : E \to \mathbb{Q}^+$. Let $r \in V$ with $d(r) = 1$. Assume that for every node $v \in V \setminus \{r\}$, the minimum capacity edge cut separating r and v has capacity of at least $d(v)$. Then, there exist polynomial time algorithms that compute

1. a packing in G of trees rooted at r such that for every node $v \in V$, the total weight of trees containing v is between $d(v)/2$ and $d(v)$, and for every edge $e \in E$, the total weight of trees containing e is at most $c(e)$; and,

2. a packing in G of trees rooted at r such that for every node $v \in V$, the total weight of trees containing v is between $d(v)$ and 1, and for every edge $e \in E$, the total weight of trees containing e is at most $2c(e)$.

The celebrated theorem of Nash-Williams (see Diestel (2000)) implies such packings in the case of $d(v) = 1$ for all $v \in V$. One of the consequences of Theorem 2 is a modification of Garg's 5-approximation for k-MST avoiding the Lagrangian relaxation. A similar modification (using another packing theorem) can be applied to the 6-approximation for metric k-median of Jain and Vazirani. It is not clear if the better algorithms that use continuity properties of the Lagrangian relaxation (for example, Garg's 3-approximation for k-MST or the 4-approximation for metric k-median of Charikar and Guha (1999)) can be derived from our packings.

The rest of the paper is organized as follows. In Section 2, we present the primal-dual algorithm for the prize collecting problem. In Section 3, we present the proof of Theorem 1. In Section 4, we give the algorithm for the quota problem. Finally, in Section 5, we give the algorithm for the budget problem.

2 The Prize Collecting Problem

In the prize collecting problem, one is given an undirected graph $G = (V, E)$ with a cost function $c : V \to \mathbb{Q}^+$, a profit function $\pi : V \to \mathbb{Q}^+$, and a specified root r. The objective is to find a subtree T of G containing the root r such that $\sum_{v \in T} c(v) + \sum_{v \notin T} \pi(v)$ is minimized. In this section we present an approximation algorithm for the prize collecting problem and analyze its approximation ratio. This algorithm is used to obtain the results in the following sections.

2.1 The Algorithm

Let $V' = V \setminus \{r\}$. Observe that the prize collecting problem can be formulated as the following integer program:

$$\begin{array}{llll}
\text{minimize} & \sum_{v\in V'} c(v)x_v + \sum_{S\subseteq V'} \pi(S) z_S & \text{subject to} & \\
& \sum_{v\in\delta(S)} x_v + \sum_{T|S\subseteq T} z_T \geq 1 & \forall S \subseteq V' & (1)\\
& x_v + \sum_{T|v\in T} z_T \geq 1 & \forall v \in V' & (2)\\
& x_v \in \{0,1\} & \forall v \in V' & \\
& z_S \in \{0,1\} & \forall S \subseteq V', &
\end{array}$$

where $x_r = 1$ and $\delta(S) = \{v \mid \exists uv \in E \text{ s.t. } u \in S, v \notin S\}$. A linear programming relaxation (denoted PC-LP) is obtained by replacing the last two sets of constraints with $x, z \geq 0$.

Consider the dual program for PC-LP, denoted by PC-D:

$$\begin{array}{lll}
\text{maximize} & \sum_{S\subseteq V'} y_S + \sum_{v\in V'} p_v & \text{subject to}\\
& \sum_{S|v\in\delta(S)} y_S + p_v \leq c(v) & \forall v \in V'\\
& \sum_{S|S\subseteq T} y_S + \sum_{v\in T} p_v \leq \pi(T) & \forall T \subseteq V'\\
& y, p \geq 0 &
\end{array}$$

The algorithm we propose for the prize collecting problem is a primal-dual algorithm which greedily constructs an implicit solution to the dual problem, PC-D.

Partition the vertices of V' into *cheap* vertices for which $c(v) \leq \pi(v)$ and *expensive* vertices, which are the rest of V'. Implicitly set $p_v = c(v)$ for each cheap v, and $p_v = \pi(v)$ for each expensive v. Consider the connected components induced by cheap vertices v and the root r. Let $\mathcal{P}$ denote the set of these components.

For each vertex $v \in V'$, define the *residual cost* $c_r(v) = c(v) - p_v$ and the *residual penalty* $\pi_r(v) = \pi(v) - p_v$. The algorithm greedily constructs a packing $\{y_S\}_{S\subseteq V'}$ which together with the values of p_v, $v \in V'$, determined above, is a feasible solution to PC-D, i.e., it satisfies:

$$\begin{array}{lll}
\sum_{S|v\in\delta(S)} y_S \leq 0 & \forall \text{cheap } v \in V' & \\
\sum_{S|v\in\delta(S)} y_S \leq c_r(v) & \forall \text{expensive } v \in V' & (3)\\
\sum_{S\subseteq T} y_S \leq \sum_{\text{cheap } v\in T} \pi_r(v) & \forall T \subseteq V' & (4)\\
y \geq 0 & &
\end{array}$$

The algorithm maintains a set $\mathcal{C}$ of *active* and *inactive* components. We say that C is *deactivated* iff it changes its status from active to inactive. Initially, $\mathcal{C} \leftarrow \mathcal{P}$, and all the components $C \in \mathcal{C}$ are active, except for the component containing the root. We raise uniformly the dual variables y_S corresponding to each active component, until either constraint (3) becomes tight for some expensive vertex v, or constraint (4) becomes tight for some active component $\tilde{C}$. In the former case, the vertex v joins a component C such that $v \in \delta(C)$, and in case there are several such components these components are merged. In the latter case, the component $\tilde{C}$ gets deactivated. The component containing the root remains inactive throughout the algorithm, and the process terminates when no active components remain. The algorithm also builds a *connection tree* T which contains edges by which expensive vertices join components. At the end of the algorithm, we perform a deletion step on the tree T. During this step, a set of vertices $S \subseteq T$ is *expendable* iff its removal does not disconnect any remaining cheap vertex from r. The code in Fig. 1 defines the algorithm more rigorously. The variables $w(C)$ maintain the values $\sum_{S \subseteq C} y_S$, and the variables $d(v)$ maintain the values $\sum_{S \ni v} y_S$. The algorithm does not need to maintain explicitly the values y_S. Their computation is included in the code for the purpose of analysis.

2.2 Analysis

Next we prove Theorem 3 that establishes an $O(\log n)$ approximation guarantee for the primal-dual algorithm.

Theorem 3.

$$\sum_{v \in T} c(v) + \beta \sum_{v \notin T} \pi(v) \leq \beta \left(\sum_{S \subseteq V'} y_S + \sum_{v \in V'} p_v \right),$$

where $\beta = O(\log n)$.

Proof. Let A be the set of cheap vertices not spanned by the final solution T, and let F denote the set of cheap vertices spanned by T. Also, let N be the set of expensive vertices spanned by T. Recall that for each cheap vertex v, $p_v = c(v)$ and for each expensive vertex v, $p_v = \pi(v)$. Therefore, to establish the claim of the theorem, it suffices to prove that the following inequality holds:

$$\sum_{v \in N} c_r(v) + \beta \sum_{v \in A} \pi_r(v) \leq \beta \sum_{S \subseteq V'} y_S,$$

where $\beta = O(\log n)$. We establish the inequality above by proving that the following two inequalities hold:

$$\sum_{v \in N} c_r(v) \quad \leq \quad \beta \sum_{S | S \bigcap F \neq \emptyset} y_S \tag{5}$$

$$\sum_{v \in A} \pi_r(v) \quad \leq \quad \sum_{S | S \bigcap F = \emptyset} y_S \tag{6}$$

First, observe that Inequality (6) follows from the greedy construction of the dual solution y by the algorithm. Indeed, consider the set $\mathcal{M}$ of maximal components deactivated by the algorithm due to the residual penalty constraints (4), such that no vertex of the component is included in the final solution T. Clearly, these components are disjoint, and as each S with non-zero value of y_S contains some cheap vertex, we get:

$$\sum_{S | S \bigcap F = \emptyset} y_S = \sum_{C \in \mathcal{M}} \sum_{S \subseteq C} y_S = \sum_{C \in \mathcal{M}} \sum_{v \in A \bigcap C} \pi_r(v) = \sum_{v \in A} \pi_r(v).$$

To see the last equality, notice that by the elimination step of the algorithm every vertex in A is included in some deactivated component not containing a vertex from F.

In the rest of the proof we show that Inequality (5) holds. We refer to vertices that caused two or more components (active or inactive) to be merged as *merge vertices*. We proceed by establishing the following properties of the dual solution constructed by the algorithm. Note that it can be easily shown by induction that $\forall v,\ d(v) = \sum_{S | v \in S} y_S$ at each step of the algorithm.

```
**PrizeCollecting**

**Initialize:**
$\mathcal{C} \leftarrow \mathcal{P}$
$y_C \leftarrow 0$ for all $C \in \mathcal{C}$
$w(C) \leftarrow 0$ for all $C \in \mathcal{C}$
$d(v) \leftarrow 0$ for all $v \in V$
All $C \in \mathcal{C}$ are active, except for $C \ni r$
$T \leftarrow$ the union of spanning trees for $C \in \mathcal{C}$

**Main loop:**
while $\exists$active $C \in \mathcal{C}$ do
    Let $d_v(C) = \max_{u \in C | \exists uv \in E} d(u)$
    Let $\epsilon_1(v) = \frac{c_r(v) - \sum_{C | v \in \delta(C)} d_v(C)}{|\{\text{active } C | v \in \delta(C)\}|}$
    Find $\tilde{v} \in V \setminus \bigcup C \in \mathcal{C}$ that minimizes $\epsilon_1 = \epsilon_1(\tilde{v})$
    Let $\epsilon_2(C) = \sum_{\text{cheap } i \in C} \pi_r(i) - w(C)$
    Find an active $\tilde{C} \in \mathcal{C}$ that minimizes $\epsilon_2 = \epsilon_2(\tilde{C})$
    $\epsilon \leftarrow \min\{\epsilon_1, \epsilon_2\}$
    $y_C \leftarrow y_C + \epsilon$ for all active $C \in \mathcal{C}$
    $w(C) \leftarrow w(C) + \epsilon$, for all active $C \in \mathcal{C}$
    $d(v) \leftarrow d(v) + \epsilon$, for all active $C \in \mathcal{C}$, $v \in C$
    if $\epsilon = \epsilon_2$ then
        Deactivate $\tilde{C}$
        Label all $v \in \tilde{C}$ by label $\tilde{C}$
    else
        Let $\overline{\mathcal{C}} = \{C \in \mathcal{C} \mid \tilde{v} \in \delta(C)\}$
        Let $\overline{C} = (\bigcup(C \in \overline{\mathcal{C}})) \bigcup \{\tilde{v}\}$
        Let $u_C = \text{argmax}\{d(w) \mid w \in C,\ w\tilde{v} \in E\}, \forall C \in \overline{\mathcal{C}}$
        $w(\overline{C}) \leftarrow \sum_{C | \tilde{v} \in \delta(C)} w(C)$
        $\mathcal{C} \leftarrow (\mathcal{C} \bigcup \{\overline{C}\}) \setminus \{C \mid \tilde{v} \in \delta(C)\}$
        if $r \in \overline{C}$ then
            $\overline{C}$ is inactive
        else
            $\overline{C}$ is active
        endif
        Add $\tilde{v}$ to the node set of $T$
        Add $u_C\tilde{v}$ to the edge set of $T$, for every $C \in \overline{\mathcal{C}}$
    endif
endwhile

**Deletion step:**
Remove from $T$ nodes that are disconnected from $r$.
while $\exists$cheap $v$ labeled $C$ s.t. $C \bigcup \delta(C)$ is expendable do
    Remove $S \bigcup \delta(S)$ from $T$
endwhile
while $\exists$expensive $v \in T$ that is expendable do
    Remove $v$ from $T$
endwhile

Output $T$.
```

Figure 1: Algorithm **PrizeCollecting** for the prize collecting problem

Lemma 4. Let $v_0, v_1, \ldots, v_k$ be a simple path in the connection tree T such that for each i, $1 \le i \le k$, v_i is not a merge vertex, and v_i gets connected to T by an edge (v_{i-1}, v_i). Then at the moment v_k gets connected to T it holds:

$$\sum_{S | v_0 \in S} y_S = c_r(v_k) + \text{dist}(v_0, v_k),$$

where dist(v_0, v_k) is the distance from v_0 to v_k in T with respect to the cost function c_r, not including the costs of v_0 and v_k.

Proof. We prove the claim of the lemma by induction on k. If $k = 1$, let $d(v_0) = \sum_{S|v_0 \in S} y_S = p$ before the increase of dual variables caused by v_1. Then $\epsilon_1 = c_r(v_1) - p$, and after the increase of dual variables by ϵ_1,

$$\sum_{S | v_0 \in S} y_S = c_r(v_1),$$

proving the basis of the induction. For the induction step, suppose that the claim of the lemma holds for vertices $v_1, \ldots, v_{i-1}$, and consider the moment when v_i gets connected to T via the edge (v_{i-1}, v_i). By the induction hypothesis, when v_{i-1} entered T, it holds that $\sum_{S|v_0 \in S} y_S = c_r(v_{i-1}) + dist(v_0, v_{i-1})$. Let $d(v_{i-1}) = q$ at the current iteration. As v_{i-1} participates only in those sets that got non-zero values after v_{i-1} entered T, it holds that

$$\sum_{S|v_0 \in S} y_S = \sum_{S|v_0 \in S, v_{i-1} \notin S} y_S + \sum_{S|v_0 \in S, v_{i-1} \in S} y_S$$

$$= c_r(v_{i-1}) + dist(v_0, v_{i-1}) + q.$$

The increase ϵ_1 in the dual variables, caused by v_k, is $c_r(v_k) - q$, and therefore, when v_k joins T, we have

$$\sum_{S|v_0 \in S} y_S = c_r(v_{i-1}) + dist(v_0, v_{i-1}) + q + c_r(v_k) - q$$

$$= dist(v_0, v_k) + c_r(v_k). \qquad \blacksquare$$

Lemma 5. Let $C_1, C_2, \ldots, C_k$ be components merged via an expensive vertex v. Let $u_i =$ argmax$\{d(w) \mid (w, v) \in E,\ w \in C_i\}$, $i = 1, \ldots, k$, that is, (u_i, v) is an edge in T by which v gets connected to C_i. Then immediately after the merge,

$$\sum_i \sum_{S|u_i \in S} y_S = c_r(v).$$

Proof. Without loss of generality, let $C_1, \ldots, C_{k'}$ be active at the time of the merge, and let $C_{k'+1}, \ldots, C_k$ be inactive. Put

$$\sum_{S|u_i \in S} y_S = q_i$$

before the merge, for each $i = 1, \ldots, k$. Then the increase of dual variables, induced by v is

$$\epsilon_1 = \frac{c_r(v) - \sum_{i=1}^k q_i}{k'}.$$

Since dual variable of each of the components $C_1, C_2, \ldots, C_{k'}$ gets increased by this amount, after the merge it holds that

$$\sum_{i=1}^k \sum_{S|u_i \in S} y_S = \sum_{i=1}^k q_i + \sum_{i=1}^{k'} \frac{c_r(v) - \sum_{j=1}^k q_j}{k'} = c_r(v). \qquad \blacksquare$$

Divide the algorithm into phases, so that at the end of each phase, the number of active components containing a cheap vertex spanned by the final solution is decreased. Let $\mathcal{N}_i$ denote the set of such components at the end of phase i of the algorithm, and let $\phi_i = |\mathcal{N}_i|$ be the number of such components at the end of phase i. Clearly, ϕ_0 is at most the number of cheap vertices, which is bounded by n. The decrease in $|\mathcal{N}_{i-1}|$ can result either from the merge of two or more components from $\mathcal{N}_{i-1} \bigcup C_r$, where C_r is the component containing the root, or from deactivation of some component from $\mathcal{N}_{i-1}$. In the former case, let $\{C_j^i\}_{j=1}^{h_i}$ be the components from $\mathcal{N}_{i-1} \bigcup C_r$ merged at the end of phase i, and let v_i be the corresponding merge vertex. In the latter case, let C^i be the component deactivated at the end of phase i, and let $h_i = 2$.

For each phase i, we bound the residual cost of a subtree D_i of T such that $\bigcup_i D_i = T$. Before defining D_i, let us consider a phase i ending in deactivating a component C^i. A vertex from C^i can appear in the final solution T if and only if the removal of C^i and $\delta(C^i)$ from T disconnects a pair of cheap vertices or a cheap vertex and the root in T. Consider the path in T between the first pair u, v of such vertices that get connected using a vertex from $C^i \bigcup \delta(C^i)$. If the removal of C^i disconnects the pair, let x and y be the first and the last vertex from C^i, respectively, along the path between u and v. If the removal of a vertex from $\delta(C^i)$ disconnects u and v, let $x \in C^i$ be the neighbor in T of the first vertex from $\delta(C_i)$ connecting u and v. We say that the vertices x, y as above become *connection* vertices at the end of phase i. Now consider the forest T constructed by the algorithm by the end of phase i. Let T_i be the subforest of T consisting of trees spanning active and connection vertices within each connected component of T. Let $D_i = T_{i+1} \setminus T_i$. If phase i ends with a merge, let $P_i \subseteq D_i$ be the set of neighbors of merge vertices in D_i, so that for every $u \in P_i$ there is a neighbor $m(u) \in D_i$ which is a merge vertex. Define $c_r(D_i) = \sum_{v \in D_i \setminus P_i} c_r(v) + \sum_{u \in P_i} \sum_{S | u \in S, m(u) \notin S} y_S$. If phase i ends in deactivating a component C^i, then if the removal of C^i disconnects a pair of active vertices with connection vertices x and y, define $c_r(D_i) = \sum_{v \in D_i} c_r(v) + \sum_{S \subseteq C^i | x \in S} y_S + \sum_{S \subseteq C^i | y \in S} y_S$. Otherwise, if the removal of a vertex from $\delta(C^i)$ disconnects a pair of active vertices, and x is a connection vertex, then define $c_r(D_i) = \sum_{v \in D_i} c_r(v) + \sum_{S \subseteq C^i | x \in S} y_S$. Note that by Lemma 5,
$\sum_{v \in N} c_r(v) = \sum_i c_r(D_i)$.

Lemma 6.

$$c_r(D_i) \leq \frac{h_i}{\phi_{i-1}} \sum_{S | S \bigcap F \neq \emptyset} y_S.$$

Proof. Let Δ denote the sum of the values of ϵ for each iteration of the algorithm, till the end of phase i. First, consider a phase i ending in a merge of active components $C_i^1, C_i^2, \ldots, C_i^{h_i}$ via a merge vertex v_i. Then D_i is a set of paths from some cheap vertex in each C_i^j to v_i, not including the paths within components inactive at the beginning of the phase i. Denote the path from an active vertex $t_j \in C_i^j$ to v_i by

$$t_j \ldots p_1, m_1, x_1 \ldots y_1, d_1 \ldots p_2, m_2, x_2 \ldots y_2, d_2 \ldots p_k, m_k,$$

$$x_k \ldots y_k, d_k \ldots p_{k+1}, v_i,$$

where m_i are merge vertices and $x_l \ldots y_l$ are paths within components inactive in the beginning of the phase i. The contribution of such path to $c_r(D_i)$ is, by Lemmas 4 and 5, at most

$$\begin{gathered}\sum_{S | t_j \in S \text{ and } p_1 \notin S} y_S + \sum_{S | p_1 \in S \text{ and } m_1 \notin S} y_S \\ + \sum_{S | m_1 \in S \text{ and } y_1 \in S} y_S + \sum_{S | d_1 \in S \text{ and } p_2 \notin S} y_S \\ + \sum_{S | p_2 \in S \text{ and } m_2 \notin S} y_S + \sum_{S | m_2 \in S \text{ and } y_2 \in S} y_S \\ \ldots \\ + \sum_{S | d_k \in S \text{ and } p_{k+1} \notin S} y_S + \sum_{S | p_{k+1} \in S \text{ and } v_i \notin S} y_S\end{gathered}$$

$$\begin{aligned} &\leq \quad \textstyle\sum_{S|t_j \in S} y_S \\ &\leq \quad \Delta \end{aligned}$$

The last inequality follows since only one component containing t_j is active at each time the dual variables are increased by a value of ϵ.

As there are h_i paths contributing to the cost of D_i, we get $c_r(D_i) \leq h_i\Delta$.

Now consider a phase i ending in deactivation of a component C^i. Then D_i is a set of at most two paths between connection vertices x, y to cheap vertices in C^i. By the same reasoning as above, each of these paths contributes at most Δ to the cost of D_i, therefore we get $c_r(D_i) \leq 2\Delta = h_i\Delta$.

Observe that each component active at the beginning of phase i must contain a vertex that has always been in some active component since the beginning of the algorithm. Clearly, for such vertex t, $\sum_{S|t\in S} y_S = \Delta$. Therefore,

$$\sum_{S|S\bigcap F\neq\emptyset} y_S \geq \phi_{i-1}\Delta.$$

Therefore, we get

$$\begin{aligned} c_r(D_i) &\leq h_i\Delta \\ &\leq \frac{h_i\Delta}{\phi_{i-1}\Delta} \sum_{S|S\bigcap F\neq\emptyset} y_S \\ &= \frac{h_i}{\phi_{i-1}} \sum_{S|S\bigcap F\neq\emptyset} y_S \qquad \blacksquare \end{aligned}$$

Let m denote the total number of phases. We can now apply a theorem of Klein and Ravi (1995) to establish that the residual cost of the subgraph $\bigcup_{i=1}^{m-1} D_i$ is at most
$O(\log n)\sum_{S|S\bigcap F\neq\emptyset} y_S$. By Lemma 6, the cost of the subgraph added at the last phase of the algorithm is at most

$$\begin{aligned} \frac{h_m}{\phi_{m-1}} \sum_{S|S\bigcap F\neq\emptyset} y_S &\leq \frac{h_m}{h_m - 1} \sum_{S|S\bigcap F\neq\emptyset} y_S \\ &\leq 2 \sum_{S|S\bigcap F\neq\emptyset} y_S. \end{aligned}$$

Therefore, Inequality 5 follows. This completes the proof of Theorem 3. $\blacksquare$

3 Tree Packings

In this section we prove Theorem 1. We prove part 1 of the theorem first, as the proof is somewhat simpler to follow. Notice that part 2 of the theorem implies part 1.

Let $V' = V \setminus \{r\}$. We wish to show that for $d : V \to \mathbb{Q}^+$ such that $d_i \leq 1$, for every $i \in V'$, $d_r = 1$, and

$$d_i \leq \sum_{j\in\delta(S)} d_j \quad \forall S \subseteq V',\ \forall i \in S, \tag{7}$$

there exists a packing $\mathcal{T}$ of connected node sets containing r, such that $T \in \mathcal{T}$ has weight λ_T in the packing, $\sum_{T \in \mathcal{T}} \lambda_T \leq 1$, and such that for every node $v \in V$, the following property holds:

$$\alpha d_v \leq \sum_{v \in V} \sum_{T \ni v} \lambda_T \leq d(v), \tag{8}$$

where $\alpha = 1/c \log |V|$. The best packing satisfying the above property is given by the solution to the following linear program which we denote CLP.

$$\begin{array}{ccc} \text{maximize} & \alpha & \text{subject to} \\ & d_i\alpha - \sum_{T \in \mathcal{T} | i \in T} \lambda_T \leq 0 & \forall i \in V \\ & \sum_{T \in \mathcal{T} | i \in T} \lambda_T \leq d_i & \forall i \in V \\ & \lambda \geq 0, & \end{array} \tag{9}$$

where $\alpha \in \mathbb{R}$ and $\lambda \in \mathbb{R}^{\mathcal{T}}$.

The dual program, which we denote by CD, is

$$\begin{array}{ccc} \text{minimize} & \sum_{i \in V} d_i x_i & \text{subject to} \\ & \sum_{i \in V} d_i y_i = 1 & \\ & \sum_{i \in T} (x_i - y_i) \geq 0 & \forall T \in \mathcal{T} \\ & x, y \geq 0, & \end{array} \tag{10}$$

where $x, y \in \mathbb{R}^V$.

Let β be the approximation guarantee for the prize collecting problem. We have $\beta = \Theta(\log n)$. Consider now a modified program, which we denote by MCD, where the constraints 10 are replaced by the constraints

$$\textstyle\sum_{i \in T} x_i - \beta \sum_{i \in T} y_i \geq 0 \quad \forall T \in \mathcal{T}. \tag{11}$$

Lemma 7. Let Z^*_{CD} and Z^*_{MCD} be the optimal values of CD and MCD, respectively (with the same coefficients d). Then, $Z^*_{\text{CD}} \geq Z^*_{\text{MCD}}/\beta$.

Proof. If (x, y) is a feasible solution to CD, then $(\beta x, y)$ is a feasible solution to MCD. ■

Let $\{\lambda_T\}_{T \in \mathcal{T}}, \alpha$ be a feasible solution to CLP. Note that $\sum_{T \in \mathcal{T}} \lambda_T \leq 1$. This follows from the fact that the root r is contained in every tree, and therefore, the inequality above is implied by constraint (9) for r.

Observe that if we could find in polynomial time a solution to CLP of value $\frac{1}{\beta}$, it would induce a packing of sets satisfying the property (8). Indeed, such packing could be obtained by picking sets T with $\lambda_T > 0$. Unfortunately, the linear program CLP has an exponential number of variables, and therefore cannot be solved in polynomial time by a linear programming algorithm. We overcome this problem by solving the modified dual program MCD. This program has an exponential number of constraints, but it can be solved in polynomial time using the ellipsoid algorithm, given a separation oracle. (Each constraint of MCD has a short description, see Grötschel, Lovász, and Schrijver (1993) for the conditions required for applying ellipsoid.) Next we describe a separation oracle for MCD, which can be applied under the assumption that $\sum_{i \in V} d_i x_i < 1$. We show that such an oracle suffices for computing a solution that suits our purpose.

Formally, under the assumption $\sum_{i \in V} d_i x_i < 1$, we wish to find a set T violating the constraint (11), i.e., a set T such that

$$\sum_{i \in T} x_i - \beta \sum_{i \in T} y_i < 0 \tag{12}$$

The following lemma provides a method for finding such a set.

Lemma 8. Let $x, y \in \mathbb{R}^V$ so that $\sum_{i \in V} d_i y_i = 1$ and $\sum_{i \in V} d_i x_i < 1$, where d is a vector satisfying the conditions of Theorem 1. Then the set T produced by **PrizeCollecting** for the instance of the prize collecting problem with $c(v) = x_v$ and $\pi(v) = y_v$, $\forall v \in V'$, satisfies Inequality (12).

Proof. By Theorem 3, **PrizeCollecting** produces a set T which satisfies:

$$\sum_{i \in T} x_i + \beta \sum_{i \notin T} y_i \leq \beta \text{PC-OPT}(x, y) \tag{13}$$

where PC-OPT(x, y) is an optimal value of PC-LP for the problem instance described above.

Define a feasible solution to PC-LP as follows. Number the vertices of $V \setminus \{r\}$ so that $d_1 \leq d_2 \leq \ldots \leq d_k$, $k = |V| - 1$. Define $S_i = \{1, 2, \ldots, i\}$, for $1 \leq i \leq k$. Set

$$\begin{aligned} x_i &= d_i, & \forall i \in V \\ z_{S_k} &= 1 - d_k, & \\ z_{S_i} &= d_{i+1} - d_i, & 1 \leq i \leq k \\ z_S &= 0, & S \neq S_i \forall i \in \{1, \ldots, k\} \end{aligned}$$

First, observe that this is, indeed, a feasible solution to PC-LP. To see that the constraints (1) are satisfied, consider any set $S \subseteq V \setminus \{r\}$. Let j be a vertex with maximal d_j in S. Then sets $T \subseteq V'$ with non-zero z_T containing S are exactly the sets $S_j, S_{j+1}, \ldots, S_k$. Therefore, we get $\sum_{T \supseteq S} z_T = 1 - d_j$. Then constraint (1) is satisfied for S as d satisfies constraint (7) for S and $j \in S$. It can be easily verified that constraints 2 are also satisfied. Now consider the value of the solution to PC-LP described above. Observe that $\sum_{S \ni i} z_S = 1 - d_i$, $1 \leq i \leq k$. Therefore, each vertex i, $1 \leq i \leq k$, contributes its penalty multiplied by $(1 - d_i)$ to the objective function value. Thus, we get that the value of this solution is $\sum_{i \in V} d_i x_i + \sum_{i \in V} (1 - d_i) y_i$. As this value is not smaller than the optimal value for PC-LP, by (13) we get

$$\begin{aligned} \sum_{i \in T} x_i + \beta \sum_{i \notin T} y_i &\leq \beta\left(\sum_{i \in V} d_i x_i + \sum_{i \in V} (1 - d_i) y_i\right) \\ &= \beta\left(\sum_{i \in V} d_i x_i + \sum_{i \in V} y_i - 1\right) \\ &< \beta \sum_{i \in V} y_i \end{aligned}$$

The latter inequality follows by the assumption that
$\sum_{i \in V} d_i x_i < 1$. Therefore, it follows that

$$\sum_{i \in T} x_i - \beta \sum_{i \in T} y_i < 0,$$

which implies the lemma. ■

We use the separation oracle described in the lemma above on the set of constraints of MCD together with the constraint $\sum_{i \in V} d_i x_i < 1$, until no feasible solution satisfying the latter constraint and the constraints already used by the algorithm can be found. At this point, the optimal value of MCD with the subset of constraints used by the ellipsoid algorithm is at least 1, and therefore the optimal value of CLP with just the variables λ_T corresponding to the used constraints is at least $\frac{1}{\beta}$. Therefore, we can solve CLP with just those variables to obtain the packing of sets satisfying (8). ■

We now proceed with the proof of part 2 of Theorem 1. The proof structure is essentially the same as the proof of part 1. However, some of the details are more complicated.

Let $V' = V \setminus \{r\}$. We show that for $d : V \to \mathbb{Q}^+$ such that $d_i \leq 1$, for every $i \in V'$, $d_r = 1$, and such that constraints (7) are satisfied, there exists a packing $\mathcal{T}$ of connected node sets containing r, such that $T \in \mathcal{T}$ has weight λ_T in the packing, $\sum_{T \in \mathcal{T}} \lambda_T \leq 1$, and such that for every node $v \in V$, the following property holds:

$$d(v) \leq \sum_{v \in V} \sum_{T \ni v} \lambda_T \leq \min\{1, d(v)\alpha\}, \tag{14}$$

where $\alpha = c \log |V|$. The best packing satisfying the above property is given by the solution to the following linear program which we denote CLP'.

$$\begin{array}{rcl} \text{minimize} & \alpha & \text{subject to} \\ & d_i\alpha - \sum_{T \in \mathcal{T} | i \in T} \lambda_T \geq 0 & \forall i \in V \\ & \sum_{T \in \mathcal{T} | i \in T} \lambda_T \geq d_i & \forall i \in V \\ & -\sum_{T \in \mathcal{T} | i \in T} \lambda_T \geq -1 & \forall i \in V \\ & \lambda \geq 0, & \end{array} \tag{15}$$

where $\alpha \in \mathbb{R}$ and $\lambda \in \mathbb{R}^{\mathcal{T}}$.

Consider the dual program for CLP', denoted CD':

$$\begin{array}{rcl} \text{maximize} & \sum_{i \in V}(d_i y_i - z_i) & \text{subject to} \\ & \sum_{i \in V} d_i x_i = 1 & \\ & \sum_{i \in T}(y_i - z_i - x_i) \geq 0 & \forall T \in \mathcal{T} \\ & x, y, z \geq 0, & \end{array} \tag{16}$$

where $x, y, z \in \mathbb{R}^V$.

We modify CD' by replacing constraints 16 by:

$$\sum_{i \in T}(\beta(y_i - z_i) - x_i) \leq 0 \quad \forall T \in \mathcal{T} \tag{17}$$

We denote the modified dual program by MCD'. By the argument, similar to that of the proof of Lemma 7,

$$Z^*_{CD'} \leq \beta Z^*_{MCD'}.$$

Next we show a separation oracle for MCD', which can be applied under assumption $\sum_{i \in V}(d_i y_i - z_i) > 1$. Formally, we wish to find a set T that violates constraint (17). Similarly to the proof of Lemma 8, we apply **PrizeCollecting** to an instance with costs x_i, and penalties y_i. Let T be the set produced by the algorithm. By the same reasoning as in the proof of Lemma 8, we get:

$$\sum_{i \in T} x_i + \beta \sum_{i \notin T} y_i \leq \beta(\sum_{i \in V} d_i x_i + \sum_{i \in V}(1 - d_i) y_i).$$

Since, by our assumption

$$\sum_{i \in V} d_i y_i > 1 + \sum_{i \in V} z_i,$$

we get:

$$\sum_{i\in T} x_i + \beta \sum_{i\notin T} y_i < \beta + \beta \sum_{i\in V} y_i - \beta - \beta \sum_{i\in V} z_i,$$

or, equivalently,

$$\sum_{i\in T} x_i - \beta \sum_{i\in T} y_i + \beta \sum_{i\in V} z_i < 0.$$

Since $z_i \geq 0, \forall i \in V$, it follows that

$$\sum_{i\in T} (x_i + \beta(z_i - y_i)) < 0,$$

implying that the set T violates constraint (17).

We use the separation oracle described above on the set of constraints of MCD' together with the constraint $\sum_{i\in V}(d_i y_i - z_i) > 1$, until no feasible solution satisfying the latter constraint, and the constraints already used by the algorithm, can be found. At this point, the optimal value of MCD' with the subset of constraints used by the ellipsoid algorithm, is at most 1, and therefore the optimal value of CLP' with just the variables λ_T corresponding to the used constraints is at most β. Therefore, we can solve CLP' with just those variables to obtain the packing of sets satisfying (14). ■

4 The Quota Problem

In the quota problem, given an undirected graph $G = (V, E)$ with a cost function $c : V \to \mathbb{Q}^+$, a profit function $\pi : V \to \mathbb{Q}^+$, a specified root r and a quota Q, the objective is to find a connected subset T of V containing r such that the total profit of T is at least Q, and the total cost of T is minimized.

For every node i, we denote by $L(i)$ the minimum cost of a path connecting r and i. Notice that in dealing with the quota problem we may eliminate nodes i with $\pi(i) \geq Q$. If the optimal solution contains such a node i, then the optimal solution is a least-cost path connecting r and i. We can compute such a path for every node i with $\pi(i) \geq Q$ and compare its cost to the solution produced by the algorithm described below. Furthermore, let Q-OPT denote the cost of the optimal solution for the quota problem. Clearly, if $L(i) >$ Q-OPT, then i is not contained in any optimal solution. We can eliminate nodes i with $L(i) >$ Q-OPT by enumerating over possible values for Q-OPT. The only interesting values for this purpose are the values $L(i)$, for all nodes i (i.e., for $L(i)$ we eliminate all nodes j with $L(j) \geq L(i)$). In the following discussion we assume that V does not contain nodes i with $\pi(i) \geq Q$ or with $L(i) >$ Q-OPT.

Consider the following linear programming relaxation for the quota problem (V' denotes $V \setminus \{r\}$), which we denote by Q-LP.

$$\begin{array}{lll}
\text{minimize} & \sum_{i\in V} c(i) d_i & \text{subject to} \\
& \sum_{i\in V} \pi(i) d_i \geq Q & \\
& d_i \leq \sum_{j\in\delta(S)} d_j & \forall S \subseteq V',\ \forall i \in S \qquad (18) \\
& d_r = 1 & \\
& d \geq 0. &
\end{array}$$

Note that constraints (18) imply $d_i \leq 1, \forall i \in V'$. Further note that Q-LP can be solved using the ellipsoid algorithm (a separation oracle requires computing minimum node cuts).

Let d be a solution to Q-LP. Let $\text{Q-LP}(d)$ denote the value of d. Consider the packing of connected sets containing r from part 2 of Theorem 1. Let $\mathcal{T}$ denote the support of the packing, and for every $T \in \mathcal{T}$, let λ_T denote the (positive) weight of T in the packing. The following lemma is a trivial consequence of Theorem 1. We omit the proof.

Lemma 9. The packing from part 2 of Theorem 1 satisfies the following conditions:

$$\sum_{T\in\mathcal{T}} \lambda_T = 1 \tag{19}$$

$$\sum_{T\in\mathcal{T}} c(T)\lambda_T \leq \beta\text{Q-LP}(d) \tag{20}$$

$$\sum_{T\in\mathcal{T}} \pi(T)\lambda_T \geq Q. \quad \blacksquare \tag{21}$$

Let $\mathcal{L} = \{T \in \mathcal{T} \mid c(T) \leq \beta\text{Q-LP}(d)\}$, $\mathcal{H} = \{T \in \mathcal{T} \mid c(T) > \beta\text{Q-LP}(d)\}$, $\mathcal{C} = \{T \in \mathcal{T} \mid \pi(T) < Q\}$. and $\mathcal{X} = \{T \in \mathcal{T} \mid \pi(T) \geq Q\}$.

Lemma 10. If $\mathcal{L} \bigcap \mathcal{X} = \emptyset$, then there exist $T_1 \in \mathcal{L}$ and $T_2 \in \mathcal{H}$ such that $\pi(T_2 \setminus T_1) \geq Q - \pi(T_1)$ and

$$\frac{c(T_2 \setminus T_1)}{\pi(T_2 \setminus T_1)} \leq \frac{\beta\text{Q-LP}(d)}{Q - p(T_1)}.$$

Proof. For $\mathcal{A} \subset \mathcal{T}$, let $\lambda_{\mathcal{A}}$ denote $\sum_{T\in\mathcal{A}} \lambda_T$. By Lemma 9, the packing is good, so by Property 20, $\mathcal{L}$ is non-empty. Let T_1 be the most profitable set in $\mathcal{L}$. Assume that $\pi(T_1) < Q$. By Property 21, $\mathcal{H}$ is non-empty. We may assume that $\mathcal{H} \bigcap \mathcal{C} = \emptyset$, otherwise remove from $\mathcal{T}$ the sets in $\mathcal{H} \bigcap \mathcal{C}$ and scale the weights of the remaining sets by $1 - \lambda_{\mathcal{H}\bigcap\mathcal{C}}$. As we eliminated sets with above-average cost and below-average profit, the modified packing is still good. Notice that $\pi(T_1)\lambda_{\mathcal{L}} \geq \sum_{T\in\mathcal{L}} \pi(T)\lambda_T$. Therefore, $\pi(T_1)\lambda_{\mathcal{L}} + \sum_{T\in\mathcal{H}} \pi(T)\lambda_T \geq Q$, or, as $\lambda_{\mathcal{L}} = 1 - \lambda_{\mathcal{H}}$,

$$\sum_{T\in\mathcal{H}} \pi(T \setminus T_1)\lambda_T \geq Q - \pi(T_1). \tag{22}$$

Notice that for all $T \in \mathcal{H}$, $\pi(T) \geq Q$ (as $\mathcal{H} \bigcap \mathcal{C} = \emptyset$), so $\pi(T \setminus T_1) \geq Q - \pi(T_1)$. Also, because the packing is good,

$$\sum_{T\in\mathcal{H}} c(T \setminus T_1)\lambda_T \leq \sum_{T\in\mathcal{H}} c(T)\lambda_T \leq \beta\text{Q-LP}(d). \tag{23}$$

It follows from Equations 22 and 23 that there exists $T_2 \in \mathcal{H}$ which, together with T_1, satisfies the claims stipulated by the lemma. $\blacksquare$

Our algorithm for the quota problem proceeds as follows. Given an optimal solution d to Q-LP, we use part 2 of Theorem 1 to compute a packing of connected sets containing r. If this packing has a set $T \in \mathcal{L} \bigcap \mathcal{X}$, we output T. Otherwise, we take $T_1 \in \mathcal{L}$ and $T_2 \in \mathcal{H}$ as exist by Lemma 10 and proceed as follows. Consider the graph induced by $T_2 \setminus T_1$. This graph is a collection of components, each of which is connected to some vertex in T_1. We take a spanning tree in each component, rooted at a vertex adjacent to a vertex of T_1. Denote the set of these spanning trees by $\mathcal{S} = \{S_1, S_2, \ldots, S_k\}$.

In what follows we describe a trimming procedure that we apply to the set $\mathcal{S}$. The procedure outputs a trimmed set of trees $\mathcal{R} = \{R_1, R_2, \ldots, R_l\}$ whose roots are taken from the set of roots of the trees in the original set. The trimmed set has the property that its total profit is at least $Q - \pi(T_1)$ and its total cost is at most $(2\beta+1)$Q-OPT. We then connect $R_1, R_2, \ldots, R_l$ to T_1. (Notice that the roots of the trimmed trees are adjacent to vertices of T_1.) Let T be the resulting tree. By the previous discussion,

$$\begin{aligned} c(T) &\leq \Theta(\log n)\text{Q-OPT} \\ \pi(T) &\geq Q \end{aligned}$$

Our algorithm outputs T.

We proceed to describe the trimming procedure. Let $p = Q - \pi(T_1)$. Notice that the total profit of the trees in $\mathcal{S}$ is at least p (because $\pi(T_2) \geq Q$). We repeatedly remove a maximal rooted subtree of any tree in $\mathcal{S}$ (including an entire tree), whose removal leaves the cost-to-profit ratio at most $\rho = \frac{\beta\text{Q-OPT}}{p}$ and the profit at least p, until no such subtree can be found. Let $\overline{\mathcal{S}}$ denote the set of remaining trees. If the profit of $\overline{\mathcal{S}}$ is at most $2p$, then its cost must be at most 2βQ-OPT. In this case we output $\mathcal{R} = \overline{\mathcal{S}}$.

Otherwise, the profit of $\overline{\mathcal{S}}$ is more than $2p$. We consider two cases.

Case 1: All rooted subtrees of trees in $\overline{\mathcal{S}}$ have cost-to-profit ratio of at most ρ. We find a subtree T' rooted at some vertex r' such that the profit of T' is at least p, but the profit of each subtree rooted at a child of r' is less than p. As the total profit of trees in $\overline{\mathcal{S}}$ is greater than $2p$, T' exists and can be found by a simple scanning procedure. Consider The subtrees rooted at the children of r'. We repeatedly remove such a subtree, until the total profit of the remaining tree T'' (including r') is between p and $2p$. Notice that by the assumption in this case, each remaining subtree has cost-to-profit ratio at most ρ. The total profit of all remaining subtrees is at most $2p$, so their total cost (excluding r') is at most $2p\frac{\beta\text{Q-OPT}}{p} = 2\beta$Q-OPT. We connect T'' to the root of the tree in $\overline{\mathcal{S}}$ containing T'' using a least-cost path in G between the two vertices. As the cost of the path is at most Q-OPT (this includes $c(r')$), the total cost of the resulting tree is at most $(2\beta+1)$Q-OPT. We output the singleton set containing this tree.

Case 2: There exists a rooted subtree T' of a tree in $\overline{\mathcal{S}}$ with cost-to-profit ratio larger than ρ. Consider a minimal T', inclusion-wise. Notice that the profit of the rest of $\overline{\mathcal{S}}$ is less than p (otherwise we would delete T'), and the cost-to-profit ratio is less than ρ. Therefore, the cost of the rest of $\overline{\mathcal{S}}$ is less than βQ-OPT. Moreover, as the total profit of $\overline{\mathcal{S}}$ is more than $2p$, the profit of T' is more than p. If T' is a single vertex r' (a leaf), then, as $c(r') \leq$ Q-OPT, $\overline{\mathcal{S}}$ has a profit of more than $2p$ and a cost of less than $(\beta+1)$Q-OPT, so we output $\mathcal{R} = \overline{\mathcal{S}}$. Otherwise, let r' be the root of T'. By the minimality assumption, every rooted subtree of T' has a cost-to-profit ratio of at most ρ. If there exists a subtree rooted at a child of r' with profit of at least p, we can apply the argument of Case 1 to this tree. Otherwise, the profit of each such subtree is less than p. We remove subtrees until the total profit (including r') is between p and $2p$. The remaining subtrees have total cost is at most βQ-OPT. We add the least-cost path in G from the root of the tree in $\overline{\mathcal{S}}$ containing T' to r'. The added cost is at most Q-OPT. We output the singleton set containing the resulting tree. ■

5 The Budget Problem

In the budget problem, given an undirected graph $G = (V, E)$ with a cost function $c : V \to \mathbb{Q}^+$, a profit function $\pi : V \to \mathbb{Q}^+$, a specified root r and a budget B, the objective is to find a subtree T of G containing r such that the total cost of T does not exceed B and the total profit of T is maximized. We may assume that for every vertex $i \in V$, $L(i) \leq B$, otherwise no feasible solution can include i, so we can discard it. Let B-OPT denote the value of the optimal solution to the budget problem.

Consider the following linear programming relaxation to the budget problem (V' denotes $V \setminus \{r\}$).

$$\text{maximize} \qquad \textstyle\sum_{i \in V} \pi(i) d_i \qquad \text{subject to}$$

$$\begin{aligned}\textstyle\sum_{i\in V} c(i)d_i \le B & \\ d_i \le \textstyle\sum_{j\in\partial S} d_j & \quad \forall S\subseteq V', \forall i\in S \\ d_r = 1 & \\ d \ge 0, &\end{aligned} \tag{24}$$

where $d \in \mathbb{R}^V$. We denote this linear program by B-LP. Note that B-LP can be solved in polynomial time using the ellipsoid algorithm.

Let d be a feasible solution to (B-LP). Let $\text{B-LP}(d)$ denote the value of d. Consider the packing of connected sets containing r from part 1 of Theorem 1. Let $\mathcal{T}$ denote the support of the packing, and for every $T \in \mathcal{T}$, let λ_T denote the weight of T in the packing. It is easy to verify that this packing satisfies the following properties:

$$\sum_{T\in\mathcal{T}} \lambda_T \le 1 \tag{25}$$

$$\sum_{v\in V} c(v) \sum_{T\ni v} \lambda_T \le B \tag{26}$$

$$\sum_{v\in V} \pi(v) \sum_{T\ni v} \lambda_T \ge \Theta\left(\frac{1}{\log n}\right) \text{B-OPT} \tag{27}$$

Next we show how the packing $\mathcal{T}$ can be used to derive an $O(\log n)$-approximation for the budget problem. Let $\mathcal{L} = \{T \in \mathcal{T} \mid c(T) \le B\}$, and $\mathcal{H} = \{T \in \mathcal{T} \mid c(T) > B\}$.

Lemma 11. At least one of the following conditions holds:

1. $\exists T \in \mathcal{L}$ such that $\pi(T) \ge \Theta(\frac{1}{\log n})\text{B-OPT}$;
2. $\exists T \in \mathcal{H}$ such that $\frac{\pi(T)}{c(T)} \ge \Theta(\frac{1}{\log n})\frac{\text{B-OPT}}{B}$.

Proof. Denote

$$\pi(\mathcal{T}) = \sum_{v\in V} \pi(v) \sum_{T\ni v} \lambda_T$$

First, consider the case when the profit from $\mathcal{L}$ is to at least half of the total profit achieved by the packing. Formally, assume

$$\sum_{v\in V} \pi(v) \sum_{T\ni v | T\in\mathcal{L}} \lambda_T \ge \frac{1}{2}\pi(\mathcal{T})$$

In this case there exists a set $T' \in \mathcal{L}$ such that

$$\sum_{v\in T'} \pi(v) \ge \frac{1}{2}\pi(\mathcal{T})$$

Indeed, assume this is not the case. Then,

$$\begin{aligned}\sum_{v\in V} \pi(v) \sum_{T\ni v | T\in\mathcal{L}} \lambda_T &= \sum_{T\in\mathcal{L}} \lambda_T \sum_{v\in T} \pi(v) \\ &< \frac{1}{2}\pi(\mathcal{T}) \sum_{T\in\mathcal{L}} \lambda_T \\ &\le \frac{1}{2}\pi(\mathcal{T}),\end{aligned}$$

contradicting our assumption. As $\pi(\mathcal{T}) = \Theta(\frac{1}{\log n})$B-OPT, the first condition of the lemma holds.

Now consider the other case when

$$\sum_{v \in V} \pi(v) \sum_{T \ni v | T \in \mathcal{L}} \lambda_T < \frac{1}{2}\pi(\mathcal{T})$$

Then,

$$\sum_{v \in V} \pi(v) \sum_{T \ni v | T \in \mathcal{H}} \lambda_T \geq \frac{1}{2}\pi(\mathcal{T})$$

Moreover, by property (26),

$$\sum_{v \in V} c(v) \sum_{T \ni v} \lambda_T \leq B,$$

and, in particular,

$$\sum_{v \in V} c(v) \sum_{T \ni v | T \in \mathcal{H}} \lambda_T \leq B$$

Therefore, we get

$$\frac{\sum_{v \in V} \pi(v) \sum_{T \ni v | T \in \mathcal{H}} \lambda_T}{\sum_{v \in V} c(v) \sum_{T \ni v | T \in \mathcal{H}} \lambda_T} \geq \frac{1}{2}\frac{\pi(\mathcal{T})}{B}$$

We conclude that there exists a set $T' \in \mathcal{H}$ such that

$$\frac{\sum_{v \in T'} \pi(v)}{\sum_{v \in T'} c(v)} \geq \frac{1}{2}\frac{\pi(\mathcal{T})}{B}.$$

Indeed, otherwise we would have

$$\begin{aligned}
\frac{\sum_{v \in V} \pi(v) \sum_{T \ni v | T \in \mathcal{H}} \lambda_T}{\sum_{v \in V} c(v) \sum_{T \ni v | T \in \mathcal{H}} \lambda_T} &= \frac{\sum_{T \in \mathcal{H}} \lambda_T \sum_{v \in T} \pi(v)}{\sum_{T \in \mathcal{H}} \lambda_T \sum_{v \in T} c(v)} \\
&< \frac{\sum_{T \in \mathcal{H}} \lambda_T \sum_{v \in T} c(v) \cdot \frac{1}{2}\frac{\pi(\mathcal{T})}{B}}{\sum_{T \in \mathcal{H}} \lambda_T \sum_{v \in T} c(v)} \\
&= \frac{1}{2}\frac{\pi(\mathcal{T})}{B}
\end{aligned}$$

By property (27), we get that the second condition of the lemma holds. ■

To obtain an $O(\log n)$-approximation for budget problem, we proceed as follows. If $\mathcal{T}$ satisfies condition 1 of Lemma 11, our algorithm outputs the set T' for which the condition holds. Otherwise, the algorithm proceeds with the set T' for which condition 2 of Lemma 11 holds. Clearly, the total profit of T' is at least $\Theta(\frac{1}{\log n})$B-OPT, but it is not a feasible solution for the budget problem, as its total cost exceeds B. Our algorithm trims T' to obtain another set T'' containing r such that

$$\frac{\sum_{v \in T''} \pi(v)}{\sum_{v \in T''} c(v)} \geq \Theta\left(\frac{1}{\log n}\right)\frac{\text{B-OPT}}{B},$$

and

$$\frac{B}{2} \leq \sum_{v \in \tilde{T}} c(v) \leq 2B.$$

A trimming procedure to obtain such a set appears in Guha, Moss, Naor, and Schieber (1999).

By the discussion above, we conclude

Theorem 12. The above algorithm is an $O(\log n)$-approximation for the budget problem. ■

6 Concluding Remarks

The packing theorems in this paper point to an interesting interplay between the primal-dual schema and rounding of linear programming relaxations. Indeed, non-constructive versions of these packing theorems and similar theorems can be deduced directly from the bounds on the dual solution cut packings underlying the related primal-dual algorithms. A better understanding of this issue is desired, and might lead to improved algorithms or new applications.

Our results for the budget problem are unsatisfactory. The problem is not known to be harder to approximate than the maximum coverage problem, for which a tight $1 - 1/e$ bound on the approximability is known. Moreover, there is no reason to believe that the problem cannot be approximated without violating the strict budget constraints. We conjecture that there is a polynomial time constant-approximation algorithm for this problem.

7 Acknowledgement

The second author would like to thank Mike Saks and David Johnson for stimulating discussions.

References

1. Arora, S., G. Karakostas. 2000. A $2 + \epsilon$ approximation algorithm for the k-MST problem. In *Proc. of the 11th SODA*.
2. Arora, S.,C. Lund. 1997. Hardness of approximations. Chapter 10 in *Approximation Algorithms for NP-Hard Problems*. PWS Publishing Company.
3. Awerbuch B., Y. Azar, A. Blum, S. Vempala. 1995. Improved approximations for minimum-weight k-trees and prize-collecting salesmen. In *Proc. of the 27th STOC*.
4. Blum A., R. Ravi, and S. Vempala. 1996. A constant factor approximation for the k-MST problem. In *Proc. of the 28th STOC*.
5. Carr R., S. Vempala. 2000. Randomized metarounding. In *Proc. of the 32nd STOC*.
6. Charikar M., S. Guha. 1999. Improved combinatorial algorithms for the facility location and k-median problems. In *Proc. of the 40th FOCS*.
7. Chudak F., T. Roughgarden, D.P. Williamson. 2001. Some notes on Garg's approximation algorithms for the k-MST problem. In *Proc. of the 12th SODA*.
8. Diestel R. 2000. *Graph Theory, 2nd Edition*. Graduate Texts in Mathematics, Volume 173, Springer-Verlag.
9. Garg N. 1996. A 3-approximation for the minimum tree spanning k vertices. In *Proc. of the 37th FOCS*.
10. Goemans M.X., D.P. Williamson. 1995. A general approximation technique for constrained forest problems. *SIAM J. Comput.* **24(2)** 296–317.
11. Grötschel M., L. Lovász, A. Schrijver. 1993. *Geometric Algorithms and Combinatorial Optimization*, second corrected edition. Springer-Verlag.
12. Guha S., A. Moss, J. Naor, B. Schieber. 1999. Efficient recovery from power outage. In *Proc. of the 13th STOC*.

13. Hochbaum D.S., ed. 1997 *Approximation Algorithms for NP-Hard Problems*. PWS Publishing Company.

14. Jain K., V.V. Vazirani. 1999. Primal-dual approximation algorithms for metric facility location and k-median problems. In *Proc. of the 40th FOCS*.

15. Johnson D.S., M. Minkoff, S. Phillips. 2000. The prize collecting Steiner tree problem: theory and practice. In *Proc. of the 11th SODA*.

16. Khuller S., A. Moss, S. Naor. 1998. The budgeted maximum coverage problem. *Information Processing Letters* **70(1)** 39–45.

17. Klein P., R. Ravi. 1995. A nearly best-possible approximation algorithm for node-weighted Steiner trees. *J. of Algorithms* **19** 104–115.