Approximating some network design problems with node costs

Guy Kortsarz
Rutgers University, Camden
guyk@camden.rutgers.edu

Zeev Nutov
The Open University of Israel
nutov@openu.ac.il

Abstract

We study several multi-criteria undirected network design problems with node costs and lengths. All these problems are related to the *Multicommodity Buy at Bulk* (MBB) problem which is as follows. We are given a graph $G = (V, E)$, demands $\{d_{st} : s, t \in V\}$, and a family $\{c_v : v \in V\}$ of subadditive cost functions. For every $s, t \in V$ we seek to send d_{st} flow units from s to t on a single path, so that $\sum_v c_v(f_v)$ is minimized, where f_v the total amount of flow through v. In the *Multicommodity Cost-Distance* (MCD) problem we are also given lengths $\{\ell(v) : v \in V\}$, and the goal is to find a subgraph H of G that minimizes $c(H)+\sum_{s,t\in V} d_{st}\cdot\ell_H(s,t)$, where $\ell_H(s,t)$ is the minimum ℓ-length of an st-path in H. The approximation for these two problems is equivalent up to a factor arbitrarily close to 2. We give an $O(\log^3 n)$-approximation algorithm for both problems for the case of demands polynomial in n. The previously best known approximation ratio was $O(\log^4 n)$ [Chekuri et. al, FOCS 2006] and [Chekuri et. al, SODA 2007].

A related problem to MBB is the *Maximum Covering Tree* (MaxCT) problem: given a graph $G = (V, E)$, costs $\{c(v) : v \in V\}$, profits $\{p(v) : v \in V\}$, and a bound C, find a subtree T of G with $c(T) \le C$ and $p(T)$ maximum. The best known approximation algorithm for MaxCT [Moss and Rabani, STOC 2001] computes a tree T with $c(T) \le 2C$ and $p(T) = \Omega(\mathsf{opt}/\log n)$. For MaxCT, we give the first real $O(\log n)$-approximation algorithm that does not violate the cost bound, and show the first nontrivial lower bound on approximation: MaxCT admits no better than $\Omega(\log\log n/\log\log\log n)$ approximation assuming NP $\not\subseteq$ Quasi(P). This solves two open questions posed in [Moss and Rabani, STOC 2001].

Another problem related to MBB is the *Shallow Light Steiner Tree* (SLST) problem, in which we are given a graph $G = (V, E)$, costs $\{c(v) : v \in V\}$, lengths $\{\ell(v) : v \in V\}$, a set $U \subseteq V$ of terminals, and a bound L. The goal is to find a subtree T of G containing U with $\mathsf{diam}_\ell(T) \le L$ and $c(T)$ minimum. We give an algorithm that computes a tree T with $c(T) = O(\log^2 n)\cdot\mathsf{opt}$ and $\mathsf{diam}_\ell(T) = O(\log n)\cdot L$. Previously, a polylogarithmic bicriteria approximation was known only for the case of edge costs and edge lengths.

Key-words: Network design, Node costs, Multicommodity Buy at Bulk, Covering tree, Approximation algorithm, Hardness of approximation.

1 Introduction

Network design problems require finding a minimum cost (sub-)network that satisfies prescribed properties, often connectivity requirements. The most fundamental problems are the ones with $0, 1$ connectivity requirements. Classic examples are: shortest path, min-cost spanning tree, min-cost Steiner tree/forest, traveling salesperson, and others. Examples of problems with high connectivity requirements are: min-cost k-flow, min-cost k-edge/node-connected spanning subgraph, Steiner network, and others. All these problems also have practical importance in applications.

Two main types of costs are considered in the literature: the edge costs and the node costs. We consider the later, which is usually more general than the edge costs variants; indeed, for most undirected network design problems, a very simple reduction transforms edge costs to node costs, but the inverse is, in general, not true. The study of network design problems with node costs is already well motivated and established from both theoretical as well as practical considerations, c.f., [20, 14, 24, 7, 6, 25]. For example, in telecommunication networks, expensive equipment such as routers and switches are located at the nodes of the underlying network, and thus it is natural to model some of these problems by assigning costs on the nodes rather than to the edges.

For some previous work on undirected network-design problems with node costs see the work of Klein and Ravi [20], Guha et al. [14], Moss and Rabani [24], and Chekuri et al. [7, 6]. We mostly focus on resolving some open problems posed in these papers.

1.1 Problems considered

Given a *length function* ℓ on the edges/nodes of a graph H, let $\ell_H(s,t)$ denote the ℓ-distance between s, t in H, that is, the minimum ℓ-length of an st-path in H (including the lengths of the endpoints). Let $\mathsf{diam}_\ell(H) = \max_{s,t \in V(H)} \ell_H(s,t)$ be the ℓ-diameter of H, that is the maximum ℓ-distance between two nodes in H. We consider the following two related problems on undirected graphs.

Multicommodity Buy at Bulk (MBB)
Instance: A graph $G = (V, E)$, a family $\{c_v : v \in V\}$ of sub-additive monotone non-decreasing cost functions, a set D of pairs from V, and positive demands $\{d_{st} : \{s,t\} \in D\}$.
Objective: Find a set $\{P_{st} : \{s,t\} \in D\}$ of st-paths so that $\sum_{v \in V} c_v(f_v)$ is minimized, where $f_v = \sum\{d_{st} : \{s,t\} \in D, v \in P_{st}\}$.

Multicommodity Cost-Distance (MCD)
Instance: A graph $G = (V, E)$, costs $\{c(v) : v \in V\}$, lengths $\{\ell(v) : v \in V\}$, a set D of pairs from V, and positive integral demands $\{d_{st} : \{s,t\} \in D\}$.
Objective: Find a subgraph H of G that minimizes

$$w(H, D) = c(H) + \sum_{\{s,t\} \in D} d_{st} \cdot \ell_H(s,t) \tag{1}$$

As linear functions are subadditive, MCD is a special case of MBB. The following statement shows that up to a factor arbitrarily close to 2, MCD and MBB are equivalent w.r.t. approximation.

Proposition 1.1 ([3]) *If there exists a ρ-approximation algorithm for* MCD *then there exists a $(2\rho + \varepsilon)$-approximation algorithm for* MBB *for any $\varepsilon > 0$.*

We consider another two fundamental problems closely related to MBB (see explanation below):

Maximum Covering Tree (MaxCT)
Instance: A graph $G = (V, E)$, costs $\{c(v) : v \in E\}$, profits $\{p(v) : v \in V\}$, and a bounds C.
Objective: Find a subtree T of G with $c(T) \leq C$ and $p(T)$ maximum.

Shallow-Light Steiner Tree (SLST)
Instance: A graph $G = (V, E)$, costs $\{c(v) : v \in V\}$, lengths $\{\ell(v) : v \in V\}$, a set $U \subseteq V$ of terminals, and a bound L.
Objective: Find a subtree T of G containing U with $\mathsf{diam}_\ell(T) \leq L$ and $c(T)$ minimum.

Each one of the problems MBB and SLST has an "edge version", where the costs/lengths are given on the edges. As was mentioned, the edge version admits an easy approximation ratio preserving reduction to the node version.

1.2 Relations between the problems considered

The common denominator of all the problems considered is the MBB problem, via the *Buy at Bulk k-Steiner Tree* ($\mathsf{BB}k\text{-}\mathsf{ST}$) problem (see [16]). We are given a graphs, G, a set of terminals $T \subseteq V$, a root r, costs and length on the edges/nodes, and an integer k. The goal is to find a tree containing r and (at least) k terminals, minimizing the cost of the tree plus the sum of distances from terminals to the root. This problem is related to MBB via the following key theorem:

Theorem 1.2 ([16]) *Let T be an optimal solution to the MBB instance at hand. Let $\mathsf{opt}_c = c(T)$ be the cost of T and let opt_ℓ be the sum of the distances from terminals in T to the root. A (ρ_c, ρ_ℓ)-bicriteria approximation algorithm for the density variant of $\mathsf{BB}k\text{-}\mathsf{ST}$ implies a solution of cost $O(\log^3 n) \cdot \rho_c \cdot \mathsf{opt}_c + O(\log n) \cdot \rho_\ell \cdot \mathsf{opt}_\ell$ for MBB.*

The algorithm that is derived from Theorem 1.2 is the only combinatorial approximation algorithm known for MBB. Improved bicriteria algorithm for $\mathsf{BB}k\text{-}\mathsf{ST}$ would imply a better combinatorial algorithm for MBB. It seems hard to improve the bicriteria approximation for $\mathsf{BB}k\text{-}\mathsf{ST}$ given in [16]. Hence we deal with the "relaxations" MaxCT and SLST of the problem, which are interesting in their own right. The hope is that they may also shed some light on $\mathsf{BB}k\text{-}\mathsf{ST}$ and thus on MBB. The MaxCT problem is similar to $\mathsf{BB}k\text{-}\mathsf{ST}$, except that we upper bound the cost and not lower bound the profit as in $\mathsf{BB}k\text{-}\mathsf{ST}$. This point is very minor and can be overcome using binary search. The main difference between $\mathsf{BB}k\text{-}\mathsf{ST}$ and MaxCT is that there are no length constraints in MaxCT. On the other hand, SLST has both costs and length. However, the constraint is on the diameter, and not on the sum of the lenghts as in $\mathsf{BB}k\text{-}\mathsf{ST}$. This point can be handled using averaging arguments. One main difference is that in SLST we must cover *all* terminals, and not only k as in $\mathsf{BB}k\text{-}\mathsf{ST}$. This difference seems quite significant and hard to handle. However, the hope is that the (separate) techniques for MaxCT and SLST can be somehow combined to derive an improved approximation for $\mathsf{BB}k\text{-}\mathsf{ST}$.

1.3 Related work

There is a vast literature on network design problems, see, e.g., [9, 11] for classic results on polynomial time algorithms. Also, see [12, 18, 29, 21, 1, 12, 19] for results on approximation algorithms.

Klein and Ravi [22] showed that the node-costs Steiner Tree problem is set-cover hard, thus it admits no $o(\log n)$ approximation unless P=NP [26]. They also obtained a matching approximation ratio using a greedy merging algorithm. Guha et al. [14] showed $O(\log n)$ integrality gap of a natural LP-relaxation for the problem. The MBB problem is motivated by economies of scale that arise in a number of applications, especially in telecommunication. The problem is studied as the fixed charge network flow problem in operations research. The first approximation algorithm for the problem is by Salman et al. [27]. For more results on the problem see [4, 13, 28, 15, 23, 8]. For the multi-commodity version MBB the first non-trivial result is due to Charikar and Karagiazova [5] who obtained an $O(\log|D|\exp(O(\sqrt{\log n \log\log n})))$-approximation, where $|D|$ is the sum of the demands. In [7] an $O(\log^4 n)$-approximation algorithm is given for the edge costs case, further generalized to the node costs case in [6]. See [2] for an $\Omega(\log^{1/2-\varepsilon} n)$-hardness result.

MaxCT was introduced in [14] motivated by efficient recovery from power outage. In [14] a pseudo approximation algorithm is presented that returns a subtree T with $c(T) \leq 2C$ and $p(T) = \Omega(P/\log^2 n)$, where P is the maximum profit under budget cost C. This was improved in [24] to produce a tree T with $c(T) \leq 2C$ and $p(T) = \Omega(P/\log n)$. For a related minimization problem when one seeks to find a minimum cost tree T with $p(T) \geq P$ [24] gives an $O(\ln n)$-approximation algorithm.

1.4 Our results

The previously best known ratio for MCD/MBB was $O(\log^4 n)$ both for edge costs [7] and node costs [6], and this was also so for polynomial demands. We improve this result by using, among other things, a better LP-relaxation for the problem.

Theorem 1.3 MCD/MBB *with polynomial demands admits an $O(\log^3 n)$-approximation algorithm.*

Our next two results are for the MaxCT problem. In the MaxCT problem it is *not* easy to meet the budget C. Indeed, all previous logarithmic approximation algorithms for the problem were in fact *pseudo-approximations*, that produced a tree of cost as high as $2C$. For MaxCT [14] gave an algorithm that computes a tree T with $c(T) \leq 2C$ and $p(T) = \Omega(P/\log^2 n)$, where $P = \max\{p(T) : c(T) \leq C, T \text{ is a subtree of } G\}$. This was improved in [24] to $c(T) \leq 2C$ and $p(T) = \Omega(P/\log n)$. We resolve two open problems posed in [24]:
1. Does MaxCT admits an $O(\log n)$-approximation algorithm without violating the cost bound?
2. Does MaxCT admits an $O(1)$ approximation algorithm?
It was conjectured in [24] that MaxCT admits an $O(\log n)$-approximation algorithm without violating the cost bound, and that it probably admits an $O(1)$ approximation ratio. We prove the first conjecture and disprove the second.

Theorem 1.4 MaxCT *admits an $O(\log n)$-approximation algorithm.*

Theorem 1.5 MaxCT *admits no constant approximation algorithm unless* $\text{NP} \subseteq \text{DTIME}(n^{O(\log n)})$. MaxCT *admits no $o(\log\log n)$ approximation algorithm unless* $\text{NP} \subseteq \text{DTIME}(n^{\text{polylog}(n)})$.

Our last result is for the SLST problem. For SLST with *edge costs* and *edge lengths*, the algorithm of [22] computes a tree T with $c(T) = O(\log n) \cdot \mathsf{opt}$ and $\mathsf{diam}_\ell(T) = O(\log n) \cdot L$. We consider the more general case of node costs and node lengths.

Theorem 1.6 SLST *with node costs and lengths admits an approximation algorithm that computes a tree T with $c(T) = O(\log^2 n) \cdot \mathsf{opt}$ and $\mathsf{diam}_\ell(T) = O(\log n) \cdot L$.*

2 Improved algorithm for MBB

In this section we prove Theorem 1.3. We give an $O(\log^2 n \cdot \log N)$-approximation algorithm for MCD with running time polynomial in N, where N is the sum of the demands plus n. If N is polynomial in n, the running time is polynomial in n, and the approximation ratio is $O(\log^3 n)$. We may assume (by duplicating nodes) that all demands are 1. Then our problem is:

Instance: A graph $G = (V, E)$, costs $\{c(v) : v \in V\}$, lengths $\{\ell(v) : v \in V\}$, a set D of node pairs.
Objective: Find a subgraph H of G that minimizes $w(H, D) = c(H) + \sum_{\{s,t\} \in D} \ell_H(s, t)$.

For the latter problem, we give an algorithm with approximation ratio $O(\log^2 n \cdot \log |D|)$. We need to assume that the *lengths* are integral and polynomial in N. This assumption is needed to solve the LP we introduce later. Specifically, we can transform our instance into a new one with lengths being integral and bounded by N^4, so that the loss in the approximation ratio is a constant, and hence is negligible in our context. By trying all nodes, we guess a node v that has the largest length among the nodes that carry a positive flow in some optimal solution H. Let $M = \ell(v)$. Zero the length of all nodes u with $\ell(u) < M/N^4$. We claim that the decrease in the length part $\sum_{\{s,t\} \in Q} \ell_H(s, t)$ of $w(H, Q)$ due to this modification is negligible. Note that the total demand that goes through one node is at most N^2. Thus if u is a node so that $\ell(u) \leq M/N^4$, then the contribution of u to the length part is at most $\ell(u) \cdot N^2 \leq M/N^2$, and the contribution of all such nodes is at most M/N. Since at least one unit of demand goes through v, $\sum_{\{s,t\} \in Q} \ell_H(s, t) \geq M$. Hence the loss is at most a fraction $1/N$ of the length part. Now, the minimum (non-zero) length of a node that contributes to the length part is at least M/N^4. Divide the length of every node by M/N^4 and round down the result to the nearest integer. It is easy to see that the rounding down only incurs a loss of factor 2 in the sum of the distances. Thus the total loss incurred is at most $2 + 1/N$, as claimed.

2.1 Approximate greedy algorithm and junction trees

We use a result about the performance of an *Approximate Greedy Algorithm* for a certain type of problems, defined as follows:

Covering Problem
Instance: A groundset Π and functions ν, w on 2^Π given by an evaluation oracle.
Objective: Find $\mathcal{P} \subseteq \Pi$ with $\nu(\mathcal{P}) = \nu(\Pi)$ and with $w(\mathcal{P})$ minimized.

A set-function f on 2^Π is *decreasing* (resp, *increasing*) if $f(\mathcal{P}_2) \leq f(\mathcal{P}_1)$ (resp., if $f(\mathcal{P}_2) \geq f(\mathcal{P}_1)$) for any $\mathcal{P}_1 \subset \mathcal{P}_2 \subseteq \Pi$, and f is *subadditive* if $f(\mathcal{P}_1 \cup \mathcal{P}_2) \leq f(\mathcal{P}_1) + f(\mathcal{P}_2)$ for all $\mathcal{P}_1, \mathcal{P}_2 \subseteq \Pi$. Given an instance of a Covering Problem, we call ν the *deficiency function* (it is assumed to be decreasing and measures how far $\mathcal{P}$ from being a feasible solution) and w the *weight function* (assumed to be increasing and subadditive). Let $\rho > 1$ and let opt be the optimal solution value for Covering Problem. The *ρ-Approximate Greedy Algorithm* starts with $\mathcal{P} = \emptyset$ and iteratively adds subsets of $\Pi - \mathcal{P}$ to $\mathcal{P}$ one after the other using the following rule. As long as $\nu(\mathcal{P}) > \nu(\Pi)$ it adds to $\mathcal{P}$ a set $\mathcal{R} \subseteq \Pi - \mathcal{P}$ so that

$$\sigma_{\mathcal{P}}(\mathcal{R}) = \frac{w(\mathcal{R})}{\nu(\mathcal{P}) - \nu(\mathcal{P} + \mathcal{R})} \leq \frac{\rho \cdot \mathsf{opt}}{\nu(\mathcal{P}) - \nu(\Pi)} \ . \tag{2}$$

The following known statement follows by a standard set-cover analysis, c.f., [20].

Theorem 2.1 *If ν is decreasing and w is increasing and subadditive, the ρ-Approximate Greedy Algorithm computes a solution $\mathcal{P}$ with $w(\mathcal{P}) \leq \rho \cdot [\ln(\nu(\emptyset) - \nu(\Pi)) + 1] \cdot \mathsf{opt}$.*

In our setting, Π is the family of all st-paths, $\{s, t\} \in D$. For a set $\mathcal{R} \subseteq \Pi$ of paths connecting a set R of pairs in D, let $\nu(\mathcal{R}) = |D| - |R|$ be the number of pairs in D not connected by paths in $\mathcal{R}$, and let $w(\mathcal{R}) = c(\mathcal{R}) + \sum_{\{s,t\} \in R} \ell(P_{st})$, where $c(\mathcal{R})$ denotes the cost of the union of the paths in $\mathcal{R}$, and P_{st} is the shortest st-path in $\mathcal{R}$. Note that $\nu(\Pi) = 0$ and $\nu(\emptyset) = |D|$. W.l.o.g., we may consider the case $\mathcal{P} = \emptyset$. Then (3) can be rewritten as:

$$\sigma(\mathcal{R}) = \frac{c(\mathcal{R})}{|R|} + \frac{\sum_{\{s,t\} \in R} \ell(P_{st})}{|R|} \leq \frac{\rho \cdot \mathsf{opt}}{|D|} \ . \tag{3}$$

The quantity $\sigma(\mathcal{R})$ in (3) is the *density* of $\mathcal{R}$; it is a sum of "cost-part" $c(\mathcal{R})/|R|$ and the remaining "length-part". The following key statement from [7] shows that with $O(\log n)$ loss in the length part of the density, we may restrict ourselves to very specific $\mathcal{R}$, as given in the following definition; in [7] it is stated for edge-costs, but the generalization to node-costs is immediate.

Definition 2.1 *A tree T with a designated node r is a* junction tree *for a subset $R \subseteq D$ of pairs in T if the unique paths in T between the pairs in R all contain r.*

Lemma 2.2 ([7], The Junction Tree Lemma) *Let H^* be an optimal solution to an* MCD *instance with $\{0,1\}$ demands. Let $C = c(H^*)$ and let $L = \sum_{\{s,t\}\in D} \ell_{H^*}(s,t)$. Then there exists a junction tree T for a subset $R \subseteq Q$ of pairs, so that $\mathsf{diam}_\ell(T) = O(\log n)\cdot L/|D|$ and $c(T)/|R| = O(C/|D|)$.*

If we could find a pair T, R as in Lemma 2.2 in polynomial time, then we would obtain an $O(\log|D| \cdot \log n)$-approximation algorithm, by Theorem 2.1. In [7] it is shown how to find such a pair that satisfy (3) with $\rho = O(\log^3 n)$. We will show how to find such a pair with $\rho = O(\log^2 n)$.

Theorem 2.3 *There exists a polynomial time algorithm that given an instance of* MCD *with $\{0,1\}$ demands computes a set $\mathcal{R}$ of paths connecting a subset $R \subseteq D$ of pairs satisfying (3) with $\rho = O(\log^2 n)$.*

Motivated by Lemma 2.2, the following LP was used in [7, 6]. Guess the common node r of the paths in $\mathcal{R}$. Let U be the union of pairs in D. Relax the integrality constraints by allowing "fractional" nodes and paths. For $v \in V$, x_v is the "fraction of v" taken into the solution. For $u \in U$, y_u is the total amount of flow v delivers to r. In the LP, we require $y_s = y_t$ for every $\{s,t\} \in D$, so $y_s = y_t$ amount of flow is delivered from s to t via r. For $u \in U$ let Π_u be the set of all ur-paths in Π, and thus $\Pi = \cup_{u\in U}\Pi_u$. For $P \in \Pi$, f_P is the amount of flow through P. Dividing all variables by $|R|$ (note that this does not affect the objective value), gives the following LP:

$$
\begin{array}{llrcll}
\text{(LP1)} & \min & \multicolumn{4}{l}{\sum_{v\in V} c(v)\cdot x_v + \sum_{P\in\Pi} \ell(P)\cdot f_P} \\
& \text{s.t.} & \sum_{u\in U} y_u & = & 1 & \\
& & \sum_{\{P\in\Pi_u | v\in P\}} f_P & \leq & x_v & v\in V,\ u\in U \\
& & \sum_{P\in\Pi_u} f_P & \geq & y_u & u\in U \\
& & y_s - y_t & = & 0 & \{s,t\}\in D \\
& & x_v, f_P, y_u & \geq & 0 & v\in V,\ P\in\Pi,\ u\in U
\end{array}
$$

2.2 The LP used

Let $A\cdot\log n\cdot L/|D|$ be the bound on the lengths of the paths in $\mathcal{R}$ guaranteed by Lemma 2.2. We use almost the same LP as (LP1), except that we seek to minimize the cost only, and restrict ourselves to paths of length at most $A\cdot\log n\cdot L/|D|$, which reflects better the statement in Lemma 2.2. For $\Pi' \subseteq \Pi$ let $\tilde{\Pi}' = \{P \in \Pi' : \ell(P) \leq A\cdot\log n\cdot L/|D|\}$. The LP we use is:

$$
\begin{array}{llrcll}
\text{(LP2)} & \min & \multicolumn{4}{l}{\sum_{v\in V} c(v)\cdot x_v} \\
& \text{s.t.} & \sum_{u\in U} y_u & = & 1 & \\
& & \sum_{\{P\in\tilde{\Pi}_u | v\in P\}} f_P & \leq & x_v & v\in V,\ u\in U \\
& & \sum_{P\in\tilde{\Pi}_u} f_P & \geq & y_u & u\in U \\
& & y_s - y_t & = & 0 & \{s,t\}\in D \\
& & x_v, f_P, y_u & \geq & 0 & v\in V,\ P\in\tilde{\Pi},\ u\in U
\end{array}
$$

Although the number of variables in (LP2) might be exponential, any basic feasible solution to (LP2) has $O(N^2)$ nonzero variables.

Lemma 2.4 *(LP2) can be solved in polynomial time (assuming the length are polynomial).*

Proof: We show that (LP2) can be solved by writing an equivalent LP with polynomial number of variables and constraints. Recall that we assume that the lengths are integral and the largest node length is at most N^4. This implies that the maximum length of a path is N^5. Define a variable $s(v,u,P,j)$ for every $v \in V$, $u \in U$, $P \in \Pi_u$, and $j \leq N^5$, so that $s(v,u,P,j) = 1$ if, and only if, v at distance j from u in the path P. Define $f(v,u,j) = \sum_{P \in \Pi_u} s(v,u,P,j) \cdot f_P$. Also $f(u,j) = \sum_v f(v,u,j)$. Thus $f(u,j)$ is the total amount of flow from u to r at distance j from u.

The above variables are used to define an alternative LP. The flow conservation inequality is

$$f(v,u,j) = \sum_{zv \in E} f(z, j - \ell(z,v), u) \quad \forall v \in V, u \in U, j \leq N^5 .$$

The flow from u to r is

$$\sum_{1 \leq j \leq N^5, vr \in E} f(u, j - \ell(vr)) .$$

It is not hard to verify, that the obtained LP is equivalent to (LP2). As its size is polynomial in N, it can be solved in time polynomial in N. Given a solution for the modified LP, the solution for (LP2) is derived by standard flow decomposition. $\square$

By Lemma 2.2 there exists a solution to (LP2) of value $O(C/|D|)$. Indeed, let $T, R, \mathcal{R}$ be as in Lemma 2.2; in particular, $c(T)/|R| = O(C/|D|)$. For $u \in T$ let P_u be the unique ur-path in T. Define a feasible solution for (LP2) as follows: $x_v = 1/|R|$ for every $v \in T$, $y_u = f_{P_u} = 1/|R|$ for every u that belongs to some pair in R, and x_u, y_u, f_P are zero otherwise. It easy to see that this solution is feasible for (LP2), and its value (cost) is $c(T)/|R| = O(C/|D|)$.

2.3 Proof of Theorem 1.3

We now proceed similarly to [7, 6]. We may assume that $\max\{1/y_u : u \in U\}$ is polynomial in n, see [6]. Partition U into $O(\log n)$ sets $U_j = \{u \in U : 1/2^{j+1} \leq y_u \leq 1/2^j\}$. There is some U_j that delivers $\Omega(1/\ln n)$ flow units to r. Focus on that U_j. Clearly, $|U_j| = \Theta(2^j)/\log n$. Setting $x'_v = \min\{\Theta(2^j) \cdot x_v, 1\}$ for all $v \in V$ and $f'_P = \min\{\Theta(2^j) \cdot f_P, 1\}$ for all $P \in \Pi$, gives a feasible solution for the following LP that requires that every node in U_j must deliver a flow unit to r.

$$\begin{array}{llrcll}
\text{(LP3)} & \min & \multicolumn{4}{l}{\sum_{v \in V} c(v) \cdot x'_v + \sum_{P \in \Pi} \ell(P) \cdot f'_P} \\
& \text{s.t.} & \sum_{\{P \in \Pi_u | v \in P\}} f'_P & \leq & x'_v & v \in V,\ u \in U_j \\
& & \sum_{P \in \Pi_u} f'_P & \geq & 1 & u \in U_j \\
& & x'_v, f'_P & \geq & 0 & v \in V,\ P \in \Pi
\end{array}$$

We bound the value of the above solution x', f' for (LP3). Since $\sum_{v \in V} c(v) x_v = O(C/|D|)$,

$$\sum_{v \in V} c(v) x'_v = O(2^j) \cdot C/|D| .$$

We later see that, since $|U_j| = \Theta(2^j/\log n)$, an extra $\log n$ factor is invoked in the cost part of our solution; if, e.g., $|U_j| = 2^j$ would hold, this $\log n$ factor would have been saved. Our main point is that the length-part of the density does not depend on the size of U_j. We show this as follows. All paths used in (LP2) are of length $O(\log n \cdot L/D)$. First, assure that $\sum_{P\in\tilde{\Pi}_u} f'_P$ is not too large. For any $u \in U_j$ the fractional values of $\{f'_P : P \in \Pi_u\}$ only affect u, namely, if $u \neq u'$ then $\tilde{\Pi}_u \cap \tilde{\Pi}_{u'} = \emptyset$. Therefore, if $\sum_{P\in\tilde{\Pi}_u} f_P >> 1$, we may assure that the sum is at most 3/2 as follows. If a single path carries at least 1/2 a unit of flow then (scaling values by only 2) this path can be used as the solution for u. Else, any minimal collection of paths delivering at least one unit of flow, deliver at most 3/2 units of flow to r. Hence the contribution of a single node u to the fractional length-part is

$$O(\log n \cdot L/|D|) \sum_{P\in\tilde{\Pi}_u} f'_P = O(\log n \cdot L/|D|) \ .$$

Over all terminals, the contribution is $O(|U_j| \cdot \log n \cdot L/|D|)$. Now, use the main theorem of [6]:

Theorem 2.5 ([6]) *There exists a polynomial time algorithm that finds an integral solution to (LP3) of value $O(\log n)$ times the optimal fractional value of (LP3).*

Hence we can find in polynomial time a tree T containing r and U_j with $c(T) = O(\log n \cdot 2^j \cdot C/|D|)$ and $\sum_{u\in U_j} \ell_T(u, r) = O(|U_j| \cdot \log^2 n \cdot L/|D|)$.

Note that if the tree contains i terminals then it contains $i/2$ pairs. This is due to the constraint $y_s = y_t$. Since the tree spans $\Theta(2^j/\log n)$ pairs, its cost-part density is $O(\log^2 n) \cdot C/|D|$. Clearly, the length-part density is $O(\log^2 n) \cdot L/|D|$. This finishes the proof of Theorem 2.3, and thus also the proof of Theorem 1.3 is complete.

3 Algorithm for MaxCT

In this section we prove Theorem 1.4. Clearly, we may assume that $c(v) \leq C$ for every $v \in V$. Let T^* be some optimal solution and let $P = p(T^*)$. We assume that P is known, as we may apply binary search and get a tight lower bound on P. The *density* of a tree T is $c(T)/p(T)$. Let T_u be a tree with a designated root u. A subtree of T_u rooted at v is the subtree induced by v and all its descendants. Let $\rho = A \ln n$ be the approximation ratio of the (bicriteria) approximation algorithm of [24], $A > 0$ is a constant. One idea of the algorithm is to guess the maximum cost node w in T^* by trying all $w \in V$ and returning the minimum cost tree over all w. Given a guess w, the algorithm executes the algorithm of [24] on a modified graph $G_w = G - \{v \in V - w : c(v) > c(w)\}$ obtained by removing all nodes of cost strictly larger than $c(w)$, and setting the cost of w to 0; all the other nodes maintain their original costs. The algorithm of [24] returns a tree T_0 with $p(T_0) \geq P/\rho$ and $c(T_0 - w) \leq 2(C - c(w))$. Eventually, we find a certain subtree $\hat{T}$ rooted at some node u. A central property is that $c(u) \leq c(w)$ and if $w \in \hat{T}$ then $c(u) = c(w)$. We show that $p(\hat{T}) = \Omega(P/\log n)$, and $c(\hat{T} - u) \leq C - c(w)$. Incorporating the cost of u we get $c(\hat{T}) \leq C - c(w) + c(u) \leq C$. Due to space limitation, see Section 6.1 for the rest of the proof of Theorem 1.4.

4 A lower bound for MaxCT

Theorem 4.1 MaxCT *admits no better than c-approximation algorithm, unless* NP $\subseteq$ DTIME$(n^{O(\ln c \cdot \exp(9c))})$.

Clearly, this implies that MaxCT admits no constant approximation algorithm unless P=NP. Also, the problem admits no $B \log\log n$-approximation for some universal constant B unless NP $\subseteq$ DTIME$(n^{\text{polylog } n})$.

Remark: The size of the instance produced is $s = n^{O(\ln c \cdot exp(9c))}$ and thus $c = \Theta(\log\log s)$. Therefore, it is not possible to get a stronger hardness than $\log\log n$ unless we get a better gap in terms of c.

4.1 The max-coverage problem

For a graph $H = (V, E)$ and $X \subseteq V$ let $\Gamma_H(X) = \Gamma_E(X) = \{v \in V - X : uv \in E \text{ for some } u \in X\}$ denote the set of *neighbors* of X in H. We say that $A' \subseteq V$ covers $B' \subseteq V$ if $B' \subseteq \Gamma_H(A')$.

Max-Coverage
Instance: A bipartite graph $H = (A + B, E)$ and a bound D with $|D| \leq |A|$ and $|B| = n$.
Objective: Find $A' \subseteq A$ with $|A'| \leq aD$ and $|\Gamma_H(A')|$ maximum.

Lemma 4.2 *Unless* NP $\subseteq$ DTIME$(n^{O(\log\log n)})$ *it is not possible to decide for any* $0 \leq a \leq \ln|B|$ *whether there is* $A' \subseteq A$ *with* $|A'| \leq aD$ *and* $|\Gamma(A')| > (1 - 1/e^{1+a})|B|$,

Proof: Let D be the minimum size of a cover of all B. By [10], the corresponding Set-Cover instance cannot be approximated better than $(1 - \varepsilon)\ln n$ for any $\varepsilon > 0$ unless NP $\subseteq$ DTIME$(n^{O(\log\log n)})$. Suppose we can find $A' \subseteq A$ with $|A'| = aD$ that covers $(1 - b)|B|$ nodes, $b \leq 1$. For the residual instance of Set-Cover, we still need to cover bn nodes in B. We can find a cover of size $D \cdot [1 + \ln(bn)]$ of the remaining nodes using the greedy algorithm. So, we can find a cover of size $D[a + 1 + \ln(bn)]$ of all B. But this cannot be smaller than $D \cdot \ln n$, unless NP $\subseteq$ DTIME$(n^{O(\log\log n)})$. So, we get that $a + 1 + \ln(bn) \geq \ln n$. This gives $b \geq e^{-(a+1)} = 1/(e^{a+1})$. □

Corollary 4.3 *Unless* NP $\subseteq$ DTIME$(n^{O(\log\log n)})$, *it is not possible to decide for any* $0 \leq a \leq \ln|B|$ *whether:*
(i) *The YES instance: There is* $A' \subseteq A$ *with* $|A'| \leq D$ *that covers all* B.
(ii) *The NO instance: there is* $A' \subseteq A$ *with* $|A'| \leq aD$ *and* $|\Gamma(A')| \geq (1 - 1/e^{1+a})|B|$.

4.2 The reduction

Define a sequence of graphs $G^1, G^2, \ldots$ by induction (see Fig. 1). To obtain G^1, take H, add a root r, and connect r to every node in A. Let $A_1 = A$ and $B_1 = B$. To obtain G^i from G^{i-1}, $i \geq 2$, take G^1 and $|B|$ copies of G^{i-1}, each corresponding to a node in B_1, and for every copy identify its root with the node corresponding to it in B_1. As the construction resembles a tree, we borrow some terms from the terminology of trees. A copy of H has *level* i if it has distance $2i - 1$ to the root r. The copies of H at level i are ordered arbitrarily. A typical copy of H at level i is denoted by $H_{ij} = (A_{ij}, B_{ij}, E_{ij})$ with i the level of the copy and j the *index* of the copy. The index j is the order statistic of the copy inside the order. Let $A^i = \bigcup_j A_{ij}$ and $B^i = \bigcup_j B_{ij}$.

An H_{ij} is an *ancestor* of a terminal y if y belongs to the subgraph rooted by some $v \in B_{ij}$; such v is called *the elements ancestor* of y in level i and is denoted $ans^i(y)$. The sets A_{ij}, B_{ij} are the ancestors sets of y in level i and are denoted by A^i_y, B^i_y. y is a *descendant* of $ans^i(y)$, A^i_y, B^i_y.

The *terminals* of G^h are $\bigcup_j B_{h,j}$, and each of them has profit 1; other nodes have profit 0. The cost of every node in A_{ij} is $1/|B|^{i-1}$ (so the nodes in $A_1 = A_{11}$ have cost 1), and the cost of any other node is 0. The cost bound is $C = h \cdot D$.

Fact 4.4 *The size (and the construction time) of the construction is* $n^{O(h)}$, *where* $n = \max\{|A|, |B|\}$.

This concludes the description of the reduction.

4.3 Analysis

While increasing the level by 1, the number Max-Coverage instances grows up by $|B|$ but the node costs go down by $|B|$. Hence the total cost of every level i is $|A|$, and the total cost of G is $h \cdot |A|$. We

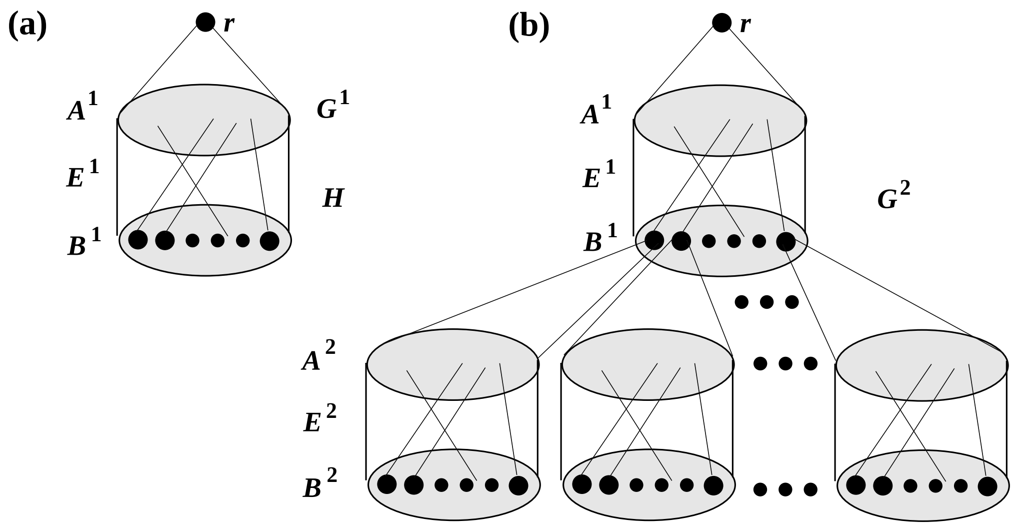

Figure 1: (a) The graph G^1. (b) The graph G^2; if instead of copies of G^1 we "attach" to nodes in B_1 roots of the copies of G^{i-1}, then we obtain G^i.

may assume that any solution T to the obtained instance of MaxCT contains r. Otherwise, we may add the shortest path from r to T; the cost added is negligible in our context. Hence a subgraph T of G is a *feasible solution* if T contains r, T is connected, and $c(T) \leq C = h \cdot D$.

Lemma 4.5 (The YES instance) *If for the* Max-Coverage *instance H there exists $A' \subseteq A$ with $|A'| = D$ that covers all B, then the obtained* MaxCT *instance G admits a feasible solution T that contains all terminals.*

Proof: Consider the graph T induced in G by r and all the copies of $A' \cup B$. This graph contains all terminals. It is easy to see that, since A' covers B, T is connected. The cost of all copies of A' at any level i is D. Summing over all levels gives total cost $c(T) = h \cdot D = C$, as claimed. $\square$

Fix a feasible solution T for MaxCT. Let c be the solution of:

$$h = 4\ln(2c) \cdot \exp(9c).$$

Our intent is to show that any solution for a no instance can cover at most $1/c$ fraction of the terminals.

Definition 4.1 *Level i in G is T-cheap if $c(T \cap A^i) < 2D$. A copy H_{ij} in level i is an T-cheap if $|T \cap A_{ij}| \leq 8 \cdot c \cdot D$. Otherwise they are expensive. A terminal y is T-expensive if at most $h/4$ of its H^i_y ancestors are T-cheap.*

In the sequel since T is fixed throughout, we use cheap instead of T-cheap. Note that for every cheap level i, by the way the cost was defined, on average $|A_{ij} \cap T| \leq 2D$. This is the reason for the definition of expensive A_{ij} above.

Claim 4.6 *At most $1/2c$ of the terminals are expensive.*

Proof: Let s be the number of cheap levels. Let i be some non-expensive level. Pick a terminal y at random. Consider the (random) ancestor H_y^i of y in level i. As y is random, by symmetry (namely, because the construction is regular) H_y^i is a random copy at level i.

As the i level is cheap, the expectation of $|T \cap A_y^i|$ is at most $2D$. Thus, with probability at least $1-1/(4c)$, $|A_y^i \cap T| \leq 8cD$. By linearity of the expectation, this implies that the expected number of levels i for which $|A_y^i \cap T| > 8 \cdot c \cdot D$ is at most $s/(4c)$. Thus, the probability that at least $s/2$ of the H_y^i in cheap levels i are expensive is at most $1/(2c)$. Therefore, with probability at least $1-1/(2c)$ at least $s/2 \geq h/4$ of the H_y^i on cheap levels are cheap. The last inequality follows as there are at least $h/2$ cheap levels. Since y was chosen at random, it implies that the fraction of expensive terminals is at most $1/2c$. □

Remove all expensive terminals. At most $1/(2c)$ fraction of the terminals are removed. Let T' be the new tree.

Lemma 4.7 (The NO instance) *If T corresponds to a no instance then T contains at most $1/c$ fraction of the terminals.*

Proof: Consider any cheap level i and a cheap H_{ij} at this level. By Corollary 4.3, for every cheap H_{ij}, as $|A'| = |A_{ij} \cap T'| \leq 8 \cdot c \cdot D$, the set A' only covers a subset $B' = \Gamma(A')$ so that

$$|B'| \leq \left(1 - \frac{1}{e^{1+8c}}\right)|B| \ .$$

This gives the same fraction of, namely, $1/e^{1+8c}$ fraction of the terminal descendants of H_{ij} that do not belong to T'. This is because every node $b \in B_{ij}$ has the same number of terminal descendants. Thus the fraction of $b \in B_{ij}$ lost (namely, that do not belong to T') equals the fraction of terminals lost. In summary, at every cheap level i, for every cheap H_{ij} a fraction of at least $1/e^{1+8c}$ of its descendants terminals do not belong to T'; indeed, $ans_y^i \notin T'$ for those terminals.

We now use Claim 4.6. Mark all H_{ij} that belong to expensive levels and also all expensive H_{ij}. By Claim 4.6, the path from every $y \in G^h \cap T'$ to r contains at least $h/4$ unmarked H_{ij} ancestors.

From this we deduce that the fraction of terminals in T' is at most:

$$\left(1 - \frac{1}{e^{9c}}\right)^{h/4} .$$

The exponent $h/4$ follows from Claim 4.6. By the definition of c we get that:

$$\left(1 - \frac{1}{2^{9c}}\right)^{h/4} < \frac{1}{2c}.$$

The fraction of terminals that belong to T is at most the number of expensive terminals plus the number of terminals of T' namely, at most $1/(2c) + 1/(2c) \leq 1/c$. The proof of Lemma 4.7 is now complete. □

Theorem 1.5 directly follows from Lemma 4.5 and Lemma 4.7.

5 Open problems

1. Does MBB with exponential demands admits an $o(\log^4 n)$ approximation ratio? What is the best approximation for the case of polynomial demands?
2. Can the approximation for SLST be improved?
3. Is MaxCT $\Omega(\log n)$ hard to approximate?

References

[1] A. Agrawal, P. Klein, and R. Ravi. When trees collide: an approximation algorithm for the generalized Steiner problem on networks. In *Proc. Symposium on the Theory of Computing (STOC)*, pages 134–144, 1991.

[2] M. Andrews. Hardness of buy-at-bulk network design. In *Proc. Symposium on the Foundations of Computer Science (FOCS)*, pages 115–124, 2004.

[3] M. Andrews and L. Zhang. Approximation algorithms for access network design. *Algorithmica*, 34(2):197–215, 2002.

[4] B. Awerbuch and Y. Azar. Buy-at-bulk network design. In *Proc. Symposium on the Foundations of Computer Science (FOCS)*, pages 542–547, 1997.

[5] M. Charikar and A. Karagiozova. On non-uniform multicommodity buy-at-bulk network design. In *Proc. Symposium on the Theory of Computing (STOC)*, pages 176–182, 2005.

[6] C. Chekuri, M. Hajiaghayi, G. Kortsarz, and M. R. Salavatipour. Polylogarithmic approximation algorithm for non-uniform multicommodity buy-at-bulk with vertex costs. In *Proc. Symposium on Discrete Algorithms (SODA)*, 2007. To appear.

[7] C. Chekuri, M. T. Hajiaghayi, G. Kortsarz, and M. R. Salavatipour. Approximation algorithms for non-uniform buy-at-bulk network design. In *Proc Symposium on the Foundations of Computer Science (FOCS)*, pages 677–686, 2006.

[8] C. Chekuri, S. Khanna, and J. Naor. A deterministic algorithm for the cost-distance problem. In *Proc. Symposium on Discrete Algorithms (SODA)*, pages 232–233, 2001.

[9] W. J. Cook, W. H. Cunningham, W. R. Pulleyblank, and A. Scrijver. *Combinatorial Optimization.* Wiley, 1998.

[10] U. Feige. A threshold of $\ln n$ for approximating set cover. *J. ACM*, 45:634–652, 1998.

[11] A. Frank. *Connectivity and network flows,* Chapter 2 in *Handbook of Combinatorics,* R. L. Graham, M. Grötschel, and L. Lovász Ed., pages 111–177. Elsvier, 1995.

[12] M. X. Goemans and D. P. Williamson. A general approximation technique for constrained forest problems. *SIAM J. Comput.*, 24(2):296–317, 1995.

[13] S. Guha, A. Meyerson, and K. Munagala. A constant factor approximation for the single sink edge installation problems. In *Proc. Symposium on the Theory of Computing (STOC)*, pages 383–388, 2001.

[14] S. Guha, A. Moss, J. S. Naor, and B. Schieber. Efficient recovery from power outage. In *Proc. Symposium on the Theory of Computing (STOC)*, pages 574–582, 1999.

[15] A. Gupta, A. Kumar, and T. Roughgarden. Simpler and better approximation algorithms for network design. In *Proc. Symposium on the Theory of Computing (STOC)*, pages 365–372, 2003.

[16] M. Hajiaghayi, G. Kortsarz, and M. R. Salavatipour. Approximating buy-at-bulk k-Steiner tree. In *Proc Workshop on Approximation algorithms (APPROX)*, pages 152–163, 2006.

[17] R. Hassin. Approximation schemes for the restricted shortest path problem. *Math. Oper. Res.*, 17(1):36–42, 1992.

[18] D. S. Hochbaum, editor. *Approximation algorithms for NP-hard problems.* PWS Publishing Co., Boston, MA, USA, 1997.

[19] K. Jain. A factor 2 approximation algorithm for the generali zed steiner network problem. *Combinatorica*, 21(1):39–60, 2001.

[20] C. Klein and R. Ravi. A nearly best-possible approximation algorithm for node-weighted steiner trees. *Journal of Algorithms*, 19(1):104–115, 1995.

[21] G. Kortsarz and Z. Nutov. *Approximating minimum cost connectivity problems,* in *Approximation Algorithms and Metaheuristics,* T. F. Gonzalez ed.,. 2007. To appear.

[22] M. V. Marathe, R. Ravi, R. Sundaram, S. S. Ravi, D. J. Rosenkrantz, and H. B. Hunt, III. Bicriteria network design problems. *J. Algorithms*, 28(1):142–171, 1998.

[23] A. Meyerson, K. Munagala, and S. Plotkin. Cost-distance: two metric network design. In *Proc. Symposium on the Foundations of Computer Science (FOCS)*, page 624, 2000.

[24] A. Moss and Y. Rabani. Approximation algorithms for constrained node weighted Steiner tree problems. In *Proc. Symposium on the Theory of Computing (STOC)*, pages 373–382, 2001.

[25] Z. Nutov. Approximating Steiner Networks with Node Weights. manuscript, 2007.

[26] R. Raz and S. Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *Proc. Symposium on the Theory of Computing (STOC)*, pages 475–484, 1997.

[27] F. S. Salman, J. Cheriyan, R. Ravi, and S. Subramanian. Approximating the single-sink link-installation problem in network design. *SIAM J. on Optimization*, 11(3):595–610, 2000.

[28] K. Talwar. The single sink buy-at-bulk LP has constant integrality gap. In *Proc. Integer Programming and Combinatorial Optimization (IPCO)*, pages 475–486, 2002.

[29] V. Vazirani. *Approximation algorithms.* Springer, 1994.

6 Appendix

6.1 Algorithm for MaxCT

6.1.1 A trimming procedure for minimally heavy trees

Definition 6.1 *A tree T is* heavy *if $p(T) > P/(8\rho)$. A tree T_u rooted at u is* minimally heavy *if it is heavy and no proper rooted subtree of T_u is heavy.*

The algorithm uses the following procedure that runs on a minimally heavy tree T_u rooted at u with $c(u) \leq c(w)$.

Procedure $MH(T_u)$

```
1. If $p(u) > P/(16\rho)$ then Return $\hat{T} = \{u\}$ and STOP.
2. $T'_u \leftarrow T_u$, $T' \leftarrow \emptyset$;
    While $p(T'_u - u) > P/(8\rho)$ do:
        $T'_u \leftarrow T'_u - T'$, where $T'$ is a subtree of $T'_u$ rooted by a child of $u$.
    EndWhile
3. Return the larger profit tree $\hat{T}$ among $T'$ (the last deleted tree) and $T'_u$.
```

Claim 6.1 *If MH is called with a minimally heavy tree T_u with $p(u) \leq P/(16\rho)$ and $c(u) \leq c(w)$ then it returns a tree $\hat{T}$ so that:*
(i) *$P/(16\rho) \leq p(\hat{T} - u) \leq P/(8\rho)$.*
(ii) *If every proper subtree of T_u has density $\leq 8\rho(C - c(w))/P$ then $c(\hat{T}) \leq C$.*

Proof: As T_u is heavy, $p(T_u) \geq P/(8\rho)$. Thus $p(u) \leq P/(16\rho)$ implies $p(T_u - u) \geq P/(16\rho)$.

Step 2 terminates because if a single child of u remains then $p(T'_u - u) \leq P/(8\rho)$ must hold since T_u is *minimally* heavy. Let T' be the last subtree removed at step 2 and let T'_u be the tree after T' is removed ($T' = \emptyset$ and $T'_u = T_u$, if no subtree is removed). Then $p(T'_u - u) + p(T') \geq P/(8\rho)$ and $p(T'_u - u) \leq P/(8\rho)$ by the stopping condition, while $p(T') \leq P/(8\rho)$ since T_u is *minimally* heavy. Part (i) of the claim follows.

We now prove Part (ii). Suppose that every proper subtree of (the initial tree) T_u has density at most $8\rho \cdot (C - c(w))/P$. Then $\max\{c(T'), c(T'_u - u)\} \leq P/(8\rho) \cdot 8\rho(C - c(w))/P = C - c(w)$. To bound the cost $c(\hat{T})$ of $\hat{T}$ in G we need (in the case $\hat{T} = T'_u$) to add the cost of u. Since $c(u) \leq c(w)$ we get $c(\hat{T}) \leq C - c(w) + c(u) \leq C$. $\square$

6.1.2 The main algorithm and its analysis

Algorithm $APPROX(G, C, P, w)$

```
1. If $p(w) > P/(2\rho)$ then return $\{w\}$ and STOP.
2. Let $T_0$ be the tree returned by the algorithm of [24] on $G_w$ with cost bound $C - c(w)$.
3. Root $T_0$ at its maximum cost node $v$ and set $T_v \leftarrow T$.
    While $T_v$ has a proper subtree $T'$ with $c(T')/p(T') > 8\rho(C - c(w))/P$ do:
        $T_v \leftarrow T_v - T'$.
    EndWhile
4. Let $T_u$ be a minimally heavy subtree of $T_v$. Return $MH(T_u)$.
```

Claim 6.2 *After step 3 ends the following holds:*
(i) *Every proper subtree of T_v has density at most $8\rho(C - c(w))/P$.*
(ii) *$p(T_v) \geq P/(4\rho)$ and thus T_v contains a minimally heavy subtree T_u.*

Proof: Part (i) is by the construction. We prove Part (ii). If w is not returned then $p(v) \leq p(w) \leq P/(2\rho)$ and thus $p(T_0 - v) \geq P/(2\rho)$. Also, $c(T_0 - v) \leq 2C - 2c(w)$. Thus, $c(T_0 - v)/p(T_0) \leq 4\rho(C - c(w))/P$. All subtrees of T_0 removed have density at least $8\rho(C - c(w))/P$. Hence, at most half the profit of $T_0 - v$ is removed. As $p(T_0 - v) \geq P/(2\rho)$, the claim follows. □

We now finish the proof of Theorem 1.4. Specifically, we claim that $APPROX$ returns a subtree with profit at least $P/(16\rho) = P/(16A \ln n)$ and cost at most C. If u is returned in step 1 of MH the claim is clear as $c(u) \leq C$. The same applies if w is returned in step 1 of $APPROX$. Else, by Claims 6.1 and 6.2 a tree of profit at least $P/(16\rho)$ and cost at most C is returned.

6.2 Algorithm for SLST

As was mentioned, for SLST with edge costs/lengths the algorithm of [22] computes a tree T with $c(T) = O(\log n) \cdot \mathsf{opt}$ and $\mathsf{diam}_\ell(T) = O(\log n) \cdot L$. This is done as follows. Let $C = \mathsf{opt}$. Maintain a disjoint partition of the terminals into pairwise disjoint *clusters*; each cluster has a unique *center*. Initialize every terminal as a cluster of size 1. Then iterate as follows. Let S be the set of cluster centers. Throughout, only terminals will be centers. For $s, t \in S$ let $c(s,t)$ be the minimum cost of an st-path among st-paths of length at most L. Although computing $c(s,t)$ is an NP-hard problem, for any $\varepsilon > 0$ we can approximate it using the FPTAS of [17], which computes a path P_{st} with $\ell(P_{st}) \leq L$ and $c(P_{st}) \leq (1+\varepsilon)c(s,t)$. Now, construct an auxiliary complete graph on S with the costs of every edge st being $c(P_{st})$. As all terminals belong to the solution, by the so called Pairing Lemma (see [22]) there exists a matching on the centers of cost at most $(1+\epsilon)C$, with at most one cluster center unmatched. Compute this perfect matching. Replace every edge st in the matching by the corresponding path P_{st} in G and merge the corresponding two clusters together, making its center to be one of s, t. At every iteration, the cost invested is at most $(1+\varepsilon)C$, the radius of each cluster is increased by at most L, and the number of clusters is roughly halved. The later implies that the number of iterations is $O(\log n)$, and the $(O(\log n), O(\log n))$ bicriteria approximation follows.

In our case of node-costs we use the decomposition of the optimum tree T^* into disjoint *spiders*. W.l.o.g., via standard reductions (see [20]), we assume that the terminals are the leaves of T^*.

Definition 6.2 *A tree T is a* spider *if it has at most one node of degree ≥ 3. A* spider decomposition *$\mathcal{D}$ of a tree T rooted at r is a collection of node-disjoint spiders, so that each of them is a rooted subtree of T, and the sets of leaves of the spiders in $\mathcal{D}$ partition the set of leaves of T.*

Lemma 6.3 ([20]) *Any tree T rooted at r admits a spider decomposition so that every spider has at least two leaves, or in the decomposition there is exactly one spider with one leaf and root r.*

The problem of finding the cheapest spider decomposition of a graph is at least as hard as the set-cover problem. Instead, we *approximate* the cost of the best spider decomposition and at the same time control the length invoked.

Lemma 6.4 *For any $\varepsilon > 0$ there exists a polynomial algorithm that computes a collection of rooted trees of total cost $O(\log n) \cdot \mathsf{opt}$ containing all terminals, so that each tree has ℓ-radius at most $(1+\varepsilon)L$, and so that every tree, except of maybe one, contains at least two terminals.*

Proof: While there are at least two terminals not belonging to any tree, iteratively find a tree F with at least 2 terminals of radius $(1+\varepsilon)L$ whose cost-density, (which is its cost over the number of terminals in it) is at most $(1+\varepsilon)$ times the one of the best spider decomposition of any optimum solution. We stress that the density in question is only with respect to non-covered terminals (only terminals not covered by previous spiders are considered). Then we remove the covered terminals, and iterate. Finding a tree of low density is done as follows. Guess the root v and the number q of terminals. For every terminal t approximate using [17], the min-cost of a v to t path of length at most L. The candidate tree for v and q is the union of the q paths from v to its best q terminals. Compute the density of the candidate tree for v, q. Then among the trees computed return one with the minimum density. The paths of the tree may intersect but since we are comparing against a spider, whose paths are node disjoint (except for the root), the resulting tree has cost density no larger than $1+\varepsilon$ times the density of the best spider. An analysis similar to the classical set-cover analysis shows that the total cost of the resulting decomposition is $O(\log n)$ times the cost of the minimum cost spider decomposition which is $O(\log n) \cdot \mathsf{opt}$. $\square$

The other parts of the algorithm are similar to the algorithm of [22] for the edge cost case. Maintain a disjoint partition of the terminals into pairwise disjoint clusters; each cluster has a unique *center*. Initialize every terminal as a cluster of size 1. Then iterate as follows. Let S be the set of cluster centers. Given $\varepsilon > 0$, compute a family of trees as in Lemma 6.4, considering only the set S of centers as terminals, and for every spider, merge the clusters of the centers it contains; the center of the new cluster is set to be one of these centers. At every iteration, the cost invested is at most $(1+\varepsilon)C \cdot O(\log n)$, the radius of each cluster is increased by at most L, and the number of clusters is roughly halved. The later implies that the number of iterations is $O(\log n)$, and the $(O(\log^2 n), O(\log n))$ bicriteria approximation follows.