

# Approximating some network design problems with node costs

Guy Kortsarz\*      Zeev Nutov†

September 9, 2009

## 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 in which 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  $t$ , so that  $\sum_v c_v(f_v)$  is minimized, where  $f_v$  the total amount of flow through  $v$ . It is shown in [2] that with a loss of  $2 - \varepsilon$  in the ratio, we may assume that each  $st$ -flow is *unsplittable*, namely, uses only one path. In the Multicommodity Cost-Distance (MCD) problem we are also given lengths  $\{\ell(v) : v \in V\}$ , and seek 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  $\ell$ -length of an  $st$ -path in  $H$ . The approximability of these two problems is equivalent up to a factor  $2 - \varepsilon$  [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 for these problems was  $O(\log^4 n)$  [Chekuri et al., FOCS 2006] and [Chekuri et al. SODA 2007].

We also consider the Maximum Covering Tree (MaxCT) problem which is closely related to MBB: 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  $c(T) \leq C$  and  $p(T)$  maximum. The best known approximation algorithm for MaxCT [Moss and Rabani, STOC 2001] computes a tree  $T$  with  $c(T) \leq 2C$  and  $p(T) = \Omega(\text{opt}/\log n)$ . We provide the first nontrivial lower bound on approximation by proving that the problem admits no better than  $\Omega(1/(\log \log n))$  approximation assuming  $\text{NP} \not\subseteq \text{Quasi(P)}$ . This holds true even if the solution is allowed to violate the budget by a constant  $\rho$ , as was done in [17] with  $\rho = 2$ . Our result disproves a conjecture of [Moss and Rabani, STOC 2001].

Another related to MBB problem 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

---

\*Rutgers University, Camden, guyk@camden.rutgers.edu. Partially supported by NSF support grant award number 0829959.

†The Open University of Israel, nutov@openu.ac.il.

$U$  with  $\text{diam}_\ell(T) \leq L$  and  $c(T)$  minimum. We give an algorithm that computes a tree  $T$  with  $c(T) = O(\log^2 n) \cdot \text{opt}$  and  $\text{diam}_\ell(T) = O(\log n) \cdot L$ . Previously, a polylogarithmic bicriteria approximation was known only for the case of edge costs and edge lengths.

# 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 latter, which is usually more general than the edge costs variants; indeed, for most undirected network design problems there is a very simple reduction that 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 [14, 11, 17, 5, 6, 18]. 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 [14], Guha et al. [11], Moss and Rabani [17], and Chekuri et al. [5, 6]. We mostly focus on resolving some open problems posed in these papers.

## 1.1 Problems considered

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

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

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

We consider two other fundamental problems closely related to MBB (see an 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  $\text{diam}_\ell(T) \leq L$  and  $c(T)$  minimum.

Each one of the above problems 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

The following problem was defined in [12]:

### $k$ -Buy at Bulk Steiner Tree ( $k$ -BBST)

*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, a root  $r \in U$ , a diameter bound  $L$ , a cost bound  $C$ , and an integer  $k$ .

*Objective:* Find a subtree  $H$  of  $G$  containing  $r$  and at least  $k - 1$  additional terminals, of diameter  $L$  and cost at most  $C$ .

This problem is related to MBB as follows (this holds for both edge and node costs). Let  $T$  be an optimal  $k$ -BBST solution for an instance at hand. Thus  $c(T) \leq C$  and  $T$  has diameter  $L$ .

**Theorem 1.2** [12] *There exist two universal constants  $c_1, c_2$  and a polynomial time algorithm that given an instance of  $k$ -BBST finds a tree  $H$  containing at least  $k/8$  terminal with cost at most  $c_1 \log^3 n \cdot c$  and diameter at most  $c_2 \cdot \log n \cdot L$ .*

In [6] the algorithm of Theorem 1.2 was used to design a *combinatorial*  $O(\log^4 n)$ -approximation algorithm for MBB. In [6] is also given an LP-based algorithm with the same ratio, which is however much slower since it repeatedly solves large linear programs. It thus seems important to try and get a better bicriteria approximation for  $k$ -BBST than the one in Theorem 1.2. Improved bicriteria algorithm for  $k$ -BBST would imply a better combinatorial algorithm for MBB. Specifically, if for  $k$ -BBST the cost returned is  $O(\rho_1) \cdot C$  and the diameter is  $O(\rho_2 \cdot L)$  then the approximation ratio derived for MBB is  $O(\log n) \cdot \rho_1 + O(\log^3 n) \cdot \rho_2$ .

Since we are facing difficulties in improving the results for  $k$ -BBST we study MaxCT and SLST which are relaxed variants of  $k$ -BBST. The hope that it may shed light on  $k$ -BBST, and thus also on MBB. Also, MaxCT and SLST are interesting in their own right.

**Comparing MaxCT and  $k$ -BBST:** MaxCT with unit costs and cost bound  $k$  is seeking a tree with  $k$  terminals and maximizing the profit. However, the big difference between  $k$ -BBST and MaxCT is that there are no length constraints in MaxCT and that the primal-dual techniques of [17] do not seem suitable to handle lengths constraints.

**Comparing SLST and  $k$ -BBST:** SLST is the particular case of  $k$ -BBST where  $k = n$ . In general, problems that seek a tree containing all terminals are usually easier than the corresponding problems that seek a tree with at least  $k$  terminals.

In summary, one may hope that techniques for MaxCT that (in the case of uniform costs) are able to find a tree with  $k$  terminals and low cost, but may not be suited to handle distances constraints, could somehow be combined with techniques that do produce a tree that is both shallow and light, but work only for the case  $k = |U|$ .

### 1.3 Related work

We survey some results on relevant network design problems with node costs. Klein and Ravi [15] showed that the **Node-Weighted Steiner Tree** problem is **Set-Cover** hard, thus it admits no  $o(\log n)$  approximation unless  $P=NP$  [19]. They also obtained a matching approximation ratio using a greedy merging algorithm. Their method was generalized in [18] to handle problems with connectivity requirements higher than 1. Guha et al. [11] 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; See [3, 9, 10, 21], and a survey in [16]. The first approximation algorithm for the problem is by Salman et al. [20]. For the multi-commodity version MBB the first non-trivial result is due to Charikar and Karagiuzova [4] who obtained an  $O(\log |D| \exp(O(\sqrt{\log n \log \log n})))$ -approximation, where  $|D|$  is the sum of

the demands. In [5] an  $O(\log^4 n)$ -approximation algorithm is given for the edge costs case, and further generalized to the node costs case in [6]. See [1] for an  $\Omega(\log^{1/2-\varepsilon} n)$ -hardness result.

The MaxCT problem was introduced in [11] motivated by efficient recovery from power outage. In [11] 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 [17] 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  $p(T) \geq P$  [17] 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 [5] and node costs [6], and this holds also 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 result is for the MaxCT problem. In [17] it is conjectured that MaxCT admits an  $O(1)$  approximation algorithm (which would have been quite helpful for dealing with  $k$ -BBST). We disprove this conjecture.

**Theorem 1.4** *Unless  $\text{NP} \subseteq \text{Quasi-P}$ , MaxCT admits no  $o(\log \log n)$ -approximation algorithm. This is so even if one is allowed to use a budget of  $\rho \cdot B$  for a constant  $\rho \geq 1$  (as was done in [17] with  $\rho = 2$ ).*

Our last result is for the SLST problem. For SLST with *edge costs* and *edge lengths*, the algorithm of [15] computes a tree  $T$  with  $c(T) = O(\log n) \cdot \text{opt}$  and  $\text{diam}_\ell(T) = O(\log n) \cdot L$ . We consider the more general case of *node costs* and *node lengths*.

**Theorem 1.5** *SLST with node costs and lengths admits an approximation algorithm that computes a tree  $T$  with  $c(T) = O(\log^2 n) \cdot \text{opt}$  and  $\text{diam}_\ell(T) = O(\log n) \cdot L$ .*

Theorems 1.3, 1.4, and 1.5 are proved in Sections 2, 3, and 4, respectively.

## 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. Let  $N$  be the new number of terminals and observe that  $N$  is polynomial in  $n$ . Then our problem is:

*Instance:* A graph  $G = (V, E)$ , costs  $\{c(v) : v \in V\}$ , lengths  $\{\ell(v) : v \in V\}$ , and a set  $D$  of node pairs.

*Objective:* Find a subgraph  $H$  of  $G$  minimizing  $w(H, D) = c(H) + \sum_{\{s,t\} \in D} \ell_H(s, t)$ .

For the latter problem, we give an  $O(\log^2 n \cdot \log |D|)$ -approximation algorithm.

## 2.1 Greedy algorithms and junction trees

We use a result about the performance of a *Greedy Algorithm* for the following type of problems:

### Covering Problem

*Instance:* A groundset  $\Pi$  and functions  $\nu, w$  on  $2^\Pi$  with  $\nu(\Pi) = 0$ .

*Objective:* Find  $\mathcal{P} \subseteq \Pi$  with  $\nu(\mathcal{P}) = \nu(\Pi)$  and with  $w(\mathcal{P})$  minimized.

Let  $\rho > 1$  and let  $\text{opt}$  be the optimal solution cost for the Covering Problem. The  $\rho$ -*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

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

The following known statement follows by a standard set-cover analysis, c.f., [14].

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

In our setting,  $\Pi$  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  $\nu(\emptyset) = |D|$ . We will show how to find such  $\mathcal{R}$  satisfying (2) with  $\rho = O(\log^2 n)$ . W.l.o.g., we may consider the case  $\mathcal{P} = \emptyset$ . Otherwise, we consider the residual instance obtained by excluding from  $D$  all pairs connected by  $\mathcal{P}$  and setting  $\mathcal{P} = \emptyset$ ; it is easy to see that if  $\mathcal{R}$  satisfies (2) for the residual instance, then this is also so for the original instance. Assuming  $\mathcal{P} = \emptyset$ , (2) can be rewritten as:

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

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 [5] 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 [5] 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 node pairs in  $T$  if the unique paths in  $T$  between the pairs in  $R$  all contain  $r$ .

**Lemma 2.2** ([5], **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  $R \subseteq D$  of pairs, so that  $\text{diam}_\ell(T) = O(\log n) \cdot L/|D|$  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 [5] it is shown how to find such a pair that satisfies (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 [5, 6]. Guess the common node  $r$  of the paths in  $\mathcal{R}$  of the junction tree  $T$ . 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  $\Pi(u)$  be the set of all  $ur$ -paths in  $\Pi$ , 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 cost), gives the following LP:

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

## 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, where  $A$  is some universal constant. 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 better reflects the statement in Lemma 2.2. Let  $\tilde{\Pi} = \{P \in \Pi : \ell(P) \leq A \cdot \log n \cdot L/|D|\}$  and let  $\tilde{\Pi}(u)$  be the paths in  $\tilde{\Pi}$  that start at  $u$ , for  $u \in U$ . Again recall that  $y_u$  is the flow delivered from  $u$  to  $r$ . The LP we use is:

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

**Lemma 2.4** *For any  $\varepsilon > 0$ , a solution for (LP2) of value  $\leq (1 + \varepsilon)\text{opt}$  can be found in time polynomial in  $n$ .*

**Proof:** The proof is similar to the proof of [8, Lemma 3.5]. Although the number of variables in (LP2) might be exponential, any basic feasible solution to (LP2) has  $O(N^2)$  non-zero variables; recall that  $N$  is the new number of nodes after the reduction to demands 1 and that  $N$  is polynomial in  $n$ . An approximate solution for (LP2) is derived from an approximate solution to its dual LP, see the proof of [8, Lemma 3.3]; if we had a polynomial time separation oracle for the dual LP, we could compute an optimal solution to (LP2) (the non-zero entries) in polynomial time. In the dual LP, the only problematic constraints are the ones that correspond to the variables  $f_P$ , which appears in the last two constraints types in (LP2), that can be rewritten as follows:

$$\begin{aligned}
\sum_{P \in \tilde{\Pi}(u)} f_P - y_u & \geq 0 & u \in U \\
x_v - \sum_{\{P \in \tilde{\Pi}(u) | v \in P\}} f_P & \geq 0 & v \in V, u \in U
\end{aligned}$$

Let  $w_u$  and  $z_{uv}$  be the corresponding dual variables. The corresponding dual constraints are:

$$w_u \leq \sum_{v \in P} z_{vu} \quad P \in \tilde{\Pi}(u).$$

To separate these constraints for a specific node  $u \in U$ , it would be sufficient to find the lightest path with respect to the weights  $z_{uv}$  among the paths in  $\tilde{\Pi}(u)$ , namely, among the  $ur$ -paths whose length with respect to  $\ell$  is at most  $A \cdot \log n \cdot L/|D|$ . This is called the **Restricted shortest Path** problem (with node costs), which is NP-hard but admits an FPTAS [13]. Therefore we get an approximate separation oracle for these constraints, which for any  $\varepsilon > 0$  checks whether there exists a path  $P \in \tilde{\Pi}(u)$  so that  $w_u \leq \sum_{v \in P} z_{vu}/(1 + \varepsilon)$ , in time polynomial in  $n$  and  $1/\varepsilon$ . The statement now follows by an argument identical to the one in the proof of [8, Lemma 3.5].  $\square$

By Lemma 2.2 there exists a solution to (LP2) of cost  $O(C/|D|)$ . 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 cost is  $c(T)/|R| = O(C/|D|)$ .

### 2.3 Proof of Theorem 1.3

We now proceed similarly to [5, 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 from every node in  $U_j$  to deliver a flow unit to  $r$ .

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

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

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

We later see that, since  $|U_j| = \Theta(2^j/\log n)$ , an extra  $\log n$  factor is invoked in the cost-density 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  $\sum_{P \in \tilde{\Pi}_u} f'_P \gg 1$ , we may assure that the sum is at most  $3/2$  as follows. If a single path carries at least  $1/2$  unit of flow then (scaling values by only 2) this path can be used as the solution for  $u$ . A collection of paths delivering a flow of value at least one from  $u$  to  $r$  is *minimal* if deleting any path from the collection brings the flow from  $u$  to  $r$  to be less than 1. Clearly,  $\tilde{\Pi}_u$  can be made minimal collection  $\tilde{\Pi}'_u$ , and we claim that  $\tilde{\Pi}'_u$  delivers at most  $3/2$  units of flow to  $r$ : removing any path  $P$  from  $\tilde{\Pi}'_u$  results in flow of value less than one. Since  $P$  carries at most  $1/2$  a unit of flow,  $\tilde{\Pi}'_u$  carries at most  $3/2$  flow units. 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 cost  $O(\log n)$  times the optimal fractional cost 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.

**Remark:** We used here the following trick. In the junction tree lemma, the length density is  $O(\log n)$  times larger than the cost density. But by the above, we were able to add  $O(\log^2 n)$  to the cost density, but only  $O(\log n)$  to the length density, making  $\rho = O(\log^2 n)$  in both quantities.

### 3 A lower bound for MaxCT

Here we prove the following statement that implies Theorem 1.4.

**Theorem 3.1** *Unless  $\text{NP} \subseteq \text{DTIME}(n^{O(\log \log n \cdot \ln c \cdot \exp(4c))})$ , MaxCT admits no better than  $c$ -approximation algorithm, even if one is allowed to use budget  $\rho \cdot B$  for a constant  $\rho \geq 1$ .*

**Remark:** The size of the instance produced is  $s = n^{O(\log \log n \log c \cdot \exp(4c))}$  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  $c$ .

We first show the result assuming the solution does not violate the budget. The extra details for the case the solution may violate the budget by a factor of  $\rho$  are given in Section 3.4.

#### 3.1 The Max-Coverage problem

For a graph  $H = (V, E)$  and  $X \subseteq V$  let  $\Gamma_H(X)$  denote the set of *neighbors* of  $X$  in  $H$ . The following is a description of the Max-Coverage problem in terms of bipartite graphs.

**Max-Coverage**

*Instance:* A bipartite graph  $H = (A + B, E)$  and an integer  $k$ .

*Objective:* Find  $A' \subseteq A$  with  $|A'| = k$  so that  $\Gamma_H(A')$  is maximum.

Here  $A$  is the collection of *sets* and  $B$  is the set of *elements*, and we say that  $A'$  covers  $B'$  if  $B' \subseteq \Gamma_H(A')$ . The following statement is very similar to [7, Proposition 5.2].

**Claim 3.2** *There exists an  $n^{O(\log \log n)}$  time reduction from any instance  $I$  of size  $n$  of an arbitrary NP-complete language to a Max-Coverage instance  $H, k$  so that the following holds.*

- *If  $I$  is a YES-instance then there exists  $A' \subseteq A$  with  $|A'| = k$  so that  $\Gamma_H(A') = B$ .*
- *If  $I$  is a NO-instance then every  $A' \subseteq A$  of size  $|A'| \leq 4c \cdot k$  covers at most  $1 - (2e)^{-4c}$  fraction of the elements, for any  $1 < c < \ln |B|$ .*

**Proof:** In the Set-Cover problem we are given a bipartite graph  $H$  as in Max-Coverage, and the goal is to find a min-size  $A' \subseteq A$  that covers all  $B$ . Feige [7] proved that there is a reduction

from any NP-complete language  $I$  of size  $n$  to a **Set-Cover** instance of size  $n^{O(\log \log n)}$  with a parameter  $k$  so that:

- If  $I$  is a YES-instance, then there exists  $A' \subseteq A$  of size  $|A'| = k$  that covers all  $B$ .
- If  $I$  is a NO-instance, then for every universal constant  $\varepsilon$ , any subset  $A' \subseteq A$  of size at most  $(1 - \varepsilon) \ln k$  does not cover all  $B$ .

Assume for the sake of contradiction that the part concerning the NO-instance in Claim 3.2 does not hold. We then show that then for  $\varepsilon = 1 - 1/(\ln(2e)) > 0$  there exists a set  $A' \subseteq A$  of size at most  $(1 - \varepsilon) \ln k$  that covers all  $B$  contradicting [7]. Apply the assumed algorithm (that follows from the negation of the NO-instance in the claim; The part about a yes instance always holds in the reduction of Feige [7]) in iterations. Each iteration adds  $4c \cdot k$  to the solution size and covers a fraction of least  $(1 - (2e)^{-4c})$  of the uncovered elements. The decline of the uncovered elements is by a fraction of  $(2e)^{-4c}$  in every iteration. Thus,  $\ell$  that satisfies  $((2e)^{-4c})^\ell = 1/n$  is an upper bound on the number of iterations needed to reduce the number of uncovered elements to 0. As

$$\ell = \frac{\ln n}{4c \cdot \ln(2e)}$$

and at every iteration  $4c \cdot k$  sets are added to the solution, the total number of sets in the solution is at most

$$\frac{1}{\ln(2e)} k = \left(1 - \left(1 - \frac{1}{\ln 2e}\right)\right) k.$$

This contradicts the result of Feige [7]. □

### 3.2 The reduction

Given a bipartite graph  $H = (A + B, E)$  that obeys Lemma 3.2 (hence has size  $n^{O(\log \log n)}$ ) define a sequence of graphs  $G^1, G^2, \dots$  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 its  $A$  sets have 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  $H_{ip} = (A_{ip}, B_{ip}, E_{ip})$  with  $i$  the level of the copy and  $p$  the *index* of the copy. This means that the  $A_{ip}$  sets are at distance  $2i - 1$  from the root and that  $p$  is the index of the copy inside level  $i$ . Let  $A^i = \bigcup_p A_{ip}$  and  $B^i = \bigcup_p B_{ip}$ .

**Definition 3.1 (The height of the reduction)**

For any constant  $c$ , let  $h = h(c) = 2e^{8c} \cdot \ln(2c)$ .

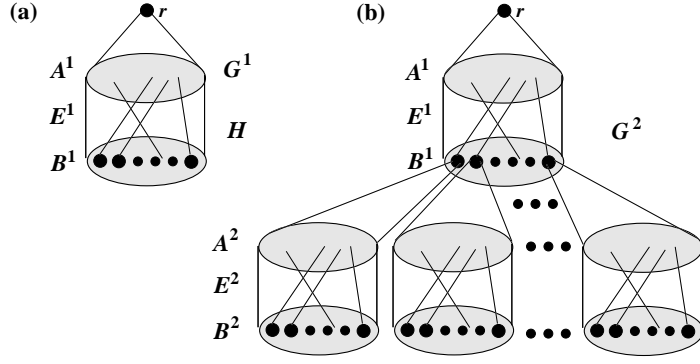


Figure 1: (a) The graph  $G^1$ . (b) The graph  $G^2$ ; if instead of copies of  $G^1$  we "attach" to nodes in  $B_1$  roots of the copies of  $G^{i-1}$ , then we obtain  $G^i$ .

**Definition 3.2** We say that a node  $v \in B_{ij}$  is an ancestor of  $H_{qp}$ ,  $q > i$ , if any path between  $r$  and an  $H_{qp}$  node must go via  $v$ . In addition, such  $H_{qp}$  are called descendants of  $v$  and are also called descendants of  $H_{ij}$ .

The terminals of  $G^h$  are  $\cup_j B_{hj}$ , namely, the elements of the last level, 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  $C = h \cdot k$ .

**Fact 3.3** The size (and the construction time) of the obtained instance of MaxCT is  $n^{h \cdot O(\log \log n)}$ .

### 3.3 Analysis

While increasing the level by 1, the number of 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 may assume that any solution  $T$  to the obtained MaxCT instance contains  $r$ . Otherwise, we may add the shortest path from  $r$  to  $T$ ; the cost added is negligible in our context.

**Lemma 3.4 (The YES-instance)** If  $I$  is a YES-instance then the obtained MaxCT instance admits a feasible subtree  $T$  of cost  $h \cdot k$ .

**Proof:** Consider the graph  $T$  induced in  $G$  by  $r$  and all the copies of  $A' \cup B$  so that  $|A'| = \text{opt}$  and  $A'$  covers  $B$ . This graph is clearly connected and contains all terminals as each  $A'$  copy covers its copy of  $B$ . The cost of all copies of  $A'$  at any level  $i$  is  $k$ . Summing over all levels gives total cost  $c(T) = h \cdot k = C$ , as claimed.  $\square$

**Lemma 3.5 (The NO-instance)** If  $I$  is a No-instance then in the obtained MaxCT instance any feasible subtree  $T$  of cost  $h \cdot k$  contains at most  $1/c$  fraction of the terminals.

In the rest of this section we prove Lemma 3.5. Fix a feasible solution  $T$  for MaxCT.

**Definition 3.3** *Level  $i$  in  $G$  is cheap (w.r.t.  $T$ ) if  $c(T \cap A^i) \leq 2k$  and is expensive otherwise. A copy  $A_{ij}$  of  $A$  on a cheap level  $i$  is called:*

- Heavy: *if  $c(T \cap A_{ij}) \geq 4 \cdot c \cdot k / |B|^{i-1}$  (namely, if  $|T \cap A_{ij}| \geq 4 \cdot c \cdot k$ ).*
- Lost: *if  $A_{ij} \cap T = \emptyset$  (because one of its ancestors  $v \in B_{pq}$  is not covered by  $T \cap A_{pq}$ ).*
- Active: *if  $A_{ij}$  is not heavy nor lost.*

Because the total budget bound is  $C = h \cdot k$  we get:

**Fact 3.6** *At most half of the levels are expensive. In a cheap level  $i$ , at most a fraction of  $1/(2c)$  (and a total of  $|B|^{i-1}/2c$ ) of the  $A$  copies are heavy.*

When we say that an  $\alpha$  fraction of the copies are active in level  $i$ , this refers to the fraction compared to the initial number of copies at level  $i$ , namely, this means that  $\alpha|B|^{i-1}$  copies are active. We now want to estimate how many copies go from active to lost going down the “tree”. We cannot estimate the loss caused by expensive levels, hence we ignore the effect of expensive levels; namely, we will assume that in each expensive level the elements of every non-lost copy may be fully covered and fully belong to  $T$ . We also assume that if  $A_{ij}$  is heavy then  $B_{ij}$  may belong to  $T$ .

Assume for the sake of contradiction that  $T$  contains at least  $1/c$  fraction of the terminals.

**Claim 3.7** *The fraction of active copies at every cheap level  $i$  is at least  $1/(2c)$ .*

**Proof:** If there is a cheap level  $i$  so that less than  $1/(2c)$  of its copies are active, then by Fact 3.6 there are less than  $2 \cdot |B|^{i-1}/(2c) = |B|^{i-1}/c$  of the  $A_{ij}$  that can belong to  $T$  (since the rest of the copies are lost copies). Since by symmetry every  $H_{ij}$  copy has the same number of descendant terminals, less than  $1/c$  fraction of the terminals belong to  $T$ ; a contradiction.  $\square$

**Claim 3.8** *If  $i$  is a cheap level then the number of active copies in level  $i + 1$  is at most  $(1 - (2e)^{-4c}/2)$  times the fraction of active copies in level  $i$ .*

**Proof:** First note that by symmetry, as there are at most  $1/(2c)$  heavy copies in level  $i$ , they can create  $1/(2c)$  fraction of active copies in level  $i + 1$ . Indeed, their children copies might not be heavy. (Of course this is an over estimation as some of the heavy copies may be lost and also even if  $A_{ij}$  is heavy it does not necessarily mean that all  $B_{ij}$  is covered and belongs to  $T$ ). However, we have also at least  $1/(2c)$  active copies in level  $i$  (see Claim 3.7). We now show that some of their children in level  $i + 1$  become lost. Fix an active copy  $A_{ij}$ . As  $A_{ij}$  is not heavy,  $|A_{ij} \cap T| \leq 4c \cdot k$ . By Claim 3.2 this implies that at least  $(2e)^{-4c}$  of the elements in  $B_{ij}$  become uncovered. Each uncovered element leads to a loss of its unique child  $A$ -copy in level  $i + 1$ . Thus the fraction of children copies of active copies lost is  $e^{-4c}$ . But the active copies may be only  $1/2$  of the total copies, since there might be as many heavy copies as active copies. Thus with respect to active copies in the best case (which is the case that all heavy copies  $A_{ij}$  were

not lost to begin with, and all  $T \cap A_{ij}$  for a heavy copy  $A_{ij}$  covers all the elements in their  $B_{ij}$ ) the fraction of active copies goes down by at least  $(2e)^{-4c}/2$  in level  $i + 1$ . The claim follows.  $\square$

**Claim 3.9** *There exists a level in which the fraction of active A-copies is strictly less than  $1/(2c)$ .*

**Proof:** Recall that  $h = 2e^{8c} \ln(2c)$  and note that  $(2e)^{4c} < e^{8c}$ . Consider the accumulative affect of going down over all cheap levels. The number of cheap levels is at least  $h/2$ . Say that at the beginning all copies are active. After going over at  $h/2$  (or more) cheap levels we get that the number of of active copies is at most

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

$\square$

The last claim contradicts Claim 3.7. This finishes the proof of Lemma 3.5, and thus the proof of Theorem 3.1 is also complete.

### 3.4 The details needed to get a bicriteria lower bound

We now show that the same hardness holds even if the NO-instance is allowed to violate the budget by a constant  $\rho$ .

Here we define  $h = 2 \cdot e^{8c\rho} \ln(2c)$ .

There is no change in the analysis of the YES-instance. A level is called cheap if the cost used in this level is at most  $2\rho \cdot k$ . An  $A_{ij}$  in a cheap level is called heavy if  $|T \cap A_{ij}| > 4c\rho \cdot k$ . The budget is still  $h \cdot k$ .

The number of cheap levels is even larger than before. The number of expensive copies at a cheap level is still at most  $1/(2c)$  fraction. This is because we have defined a copy as heavy if  $|T \cap A_{ij}| \geq 4c\rho \cdot k$ , and a cheap level has cost at most  $2\rho \cdot k$ .

Note that now we may assume that a light copy contains at most  $4c\rho \cdot k$  sets. Thus the fraction of remaining active copies goes down by at least  $(1 - e^{-4\rho \cdot c \cdot \text{opt}}/2)$  fraction at every cheap level. However, since we defined  $h = 2 \cdot e^{8c\rho} \cdot \ln 2c$  the affect of the above is canceled by the larger  $h$ .

## 4 Algorithm for SLST

As was mentioned, for SLST with edge costs/lengths the algorithm of [15] computes a tree  $T$  with  $c(T) = O(\log n) \cdot \text{opt}$  and  $\text{diam}_\ell(T) = O(\log n) \cdot L$ . This is done as follows. Let  $C = \text{opt}$ . 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 [13], 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 [15]) there exists a matching on the centers of cost at most  $(1 + \varepsilon)C$ , 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 [14]), we assume that the terminals are the leaves of  $T^*$ .

**Definition 4.1** *A tree is a spider if it has at most one node of degree  $\geq 3$ . 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 so that the leaf sets of the spiders in  $\mathcal{D}$  partition the leaf set of  $T$ .*

**Lemma 4.1** ([14]) *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 4.2** *For any  $\varepsilon > 0$  there exists a polynomial algorithm that computes a collection of rooted trees of total cost  $O(\log n) \cdot \text{opt}$  containing all terminals, so that each tree has  $\ell$ -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 [13], 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  $O(\log n) \cdot \text{opt}$ .  $\square$

The other parts of the algorithm are similar to the algorithm of [15] 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 4.2, 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.

## References

- [1] M. Andrews. Hardness of buy-at-bulk network design. In *FOCS*, pages 115–124, 2004.
- [2] M. Andrews and L. Zhang. Approximation algorithms for access network design. *Algorithmica*, 34(2):197–215, 2002.
- [3] M. Balinski. Fixed cost transportation problems. *Nav. Res. Log. Quart.*, pages 58–76, 1961.
- [4] M. Charikar and A. Karagiozova. On non-uniform multicommodity buy-at-bulk network design. In *STOC*, pages 176–182, 2005.
- [5] C. Chekuri, M. T. Hajiaghayi, G. Kortsarz, and M. R. Salavatipour. Approximation algorithms for non-uniform buy-at-bulk network design. In *FOCS*, pages 677–686, 2006.
- [6] C. Chekuri, M. T. Hajiaghayi, G. Kortsarz, and M. R. Salavatipour. Approximation algorithms for node-weighted buy-at-bulk network design. In *SODA*, pages 1265–1274, 2007.
- [7] U. Feige. A threshold of for approximating set cover. *J. ACM*, 45:634–652, 1998.
- [8] M. Feldman, G. Kortsarz, and Z. Nutov. Improved approximating algorithms for directed dteiner forest. In *SODA*, pages 922–931, 2009.
- [9] G.Gallo, C. Sandi, and C. Sodini. An algorithm for the min concave cost flow problem. *Euro. Journal on Operation Research*, 4:248–255, 1980.

- [10] M. Goldstein and B. Rothfrab. The one terminal telepak problem. *Operation research*, 19:156–169, 1971.
- [11] S. Guha, A. Moss, J. S. Naor, and B. Schieber. Efficient recovery from power outage. In *STOC*, pages 574–582, 1999.
- [12] M. T. Hajiaghayi, G. Kortsarz, and M. R. Salavatipour. Approximating buy-at-bulk  $k$ -Steiner tree. In *APPROX*, pages 152–163, 2006.
- [13] R. Hassin. Approximation schemes for the restricted shortest path problem. *Mathematics of Operations Research*, 17(1):36–42, 1992.
- [14] C. Klein and R. Ravi. A nearly best-possible approximation algorithm for node-weighted steiner trees. *Journal of Algorithms*, 19(1):104–115, 1995.
- [15] 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.
- [16] M. Minoux. Network syntesis and optimum network design problems: Models, solution methods and applications. *Networks*, 19:313–360, 1989.
- [17] A. Moss and Y. Rabani. Approximation algorithms for constrained node weighted Steiner tree problems. In *STOC*, pages 373–382, 2001.
- [18] Z. Nutov. Approximating steiner networks with node weights. In *LATIN*, pages 411–422, 2008.
- [19] R. Raz and S. Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *STOC*, pages 475–484, 1997.
- [20] 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.
- [21] W. Zangwill. Minimum concave flow in certain networks. *Mgmt Sci.*, 14:429–450, 68.