Approximation Algorithms for Node-Weighted Buy-at-Bulk Network Design

C. Chekuri* M. T. Hajiaghayi† G. Kortsarz‡ M. R. Salavatipour§

Abstract

We present algorithms with poly-logarithmic approximation ratios for the buy-at-bulk network design problem in the node-weighted setting. We obtain the following results where h is the number of pairs in the input.

- An $O(\log h)$ approximation for the single-sink non-uniform buy-at-bulk network design. Unless $P = NP$ this ratio is tight up to constant factors.
- An $O(\log^4 h)$ approximation for the multi-commodity non-uniform buy-at-bulk network design problem.

1 Introduction

Network design problems involve finding a minimum cost (sub) network that satisfies various properties, often involving connectivity. Simple examples include spanning trees, Steiner trees, and k-connected subgraphs. These problems are of fundamental importance in combinatorial optimization and also arise in a number of applications in computer science and operations research. The cost in a typical network design problem is some function of the chosen *edges*. In this paper we consider network design problems with costs (or weights) on both *edges* and *nodes* of the graph. We are motivated by both theoretical as well as practical considerations. The node-weighted problems are a natural generalization of the edge weighted problems (weights on edges can be translated to weights on nodes in an easy fashion). It is often possible to reduce the node-weighted problem to a corresponding edge-weighed problem, however this requires making the graph directed. Problems on directed graphs are typically more complex (harder to approximate for instance) than the ones in undirected graphs and hence it is desirable to work directly on node-weighted problems in undirected graphs. Node weights also arise naturally in a number of practical applications. For example, in telecommunications, expensive equipment such as routers and switches are at the nodes of the underlying network and it is natural to model some of these problems as node-weighted problems. Often, these node weights are translated into edge weights in an approximate fashion since the edge-weighted problems are better understood. Klein and Ravi [24] explored node-weighted network design and gave an $O(\log h)$ approximation algorithm for the node-weighted Steiner tree problem (h is the number of terminals). They also showed that their approach yields the same approximation ratio for the class of all $0, 1$ proper functions and in particular for the Steiner forest problem. In some subsequent work, Guha and Khuller [14], Guha et al. [15] and Moss and Rabani [26] considered variants of the node-weighted Steiner tree problem. Apart from these few papers the literature on node-weighted problems is sparse in comparison to the work on edge-weighted problems. We remark that some of the work on network design problem for node and element connectivity [12] is also in the edge-weight model. Node-capacitated routing problems (see e.g. [8, 11, 19] for some recent works) are also studied but those usually do not involve cost minimization.

In this paper we consider the node-weighted versions of the buy-at-bulk network design problem. These problems are motivated by economies of scale that arise in a number of applications, most notably in telecommunications. These are studied as fixed charge network flow problems in operations research. Approximation algorithms, starting with the work of Salman et al. [27], have been of much interest. All the known results are for the edge-weighted problems. In this paper we obtain the first non-trivial approximation algorithms for the node-weighted versions of these problems. We give a formal description of the problem before describing our results and related work.

Node-weighted Non-uniform Buy-at-Bulk: We are given an undirected graph G and node pairs $s_1t_1, s_2t_2, \ldots, s_ht_h$. The pairs have non-negative demands; δ_i is the demand for pair s_it_i. Each node $v \in V$ has a monotone sub-additive real valued function

*Department of Computer Science, University of Illinois, Urbana, IL 61801. Email: chekuri@cs.uiuc.edu. Part of this work was done while the author was at Lucent Bell Labs.

†Department of Computer Science, Carnegie Mellon University. Email: hajiagha@cs.cmu.edu.

‡Department of Computer Science, Rutgers University-Camden. Email: guyk@crab.rutgers.edu.

§Department of Computing Science, University of Alberta. Email: mreza@cs.ualberta.ca. Supported by NSERC grant No. G121210990, and a faculty start-up grant from University of Alberta.

$f_v : \mathcal{R}^+ \rightarrow \mathcal{R}^+$ associated with it. A feasible solution consists of paths $P_1, P_2, \ldots, P_h$ such that P_i connects s_i and t_i. Given the paths, δ_i flow is routed along P_i for $1 \leq i \leq h$. The cost of the flow is $\sum_{v \in V} f_v(x_v)$ where x_v is the total flow that is routed through a node v, namely, $x_v = \sum_{i | v \in P_i} \delta_i$. We assume that a flow that originates at a node v is also routed through v. The objective is to find a feasible solution (or routing) for the pairs that minimizes the total cost. (A relaxation of the problem would allow the flow for each pair to be split among multiple paths. However this does not reduce the cost for the class of sub-additive functions under consideration.) We obtain a single-sink (or equivalently single-source) problem if all pairs share a common sink node, that is, the pairs are of the form $st_1, st_2, \ldots, st_k$ with s as the sink. We obtain a *uniform* instance if there is a function f such that for each v, $f_v = c(v)f$ for some $c : V \rightarrow \mathcal{R}^+$.

Note that the node-weighted Steiner tree and the node-weighted Steiner forest problems are special cases of the (uniform) single-sink and multi-commodity problems respectively. A simple reduction from the set cover problem [24] shows that the node-weighted Steiner tree problem is hard to approximate to within a factor of $\Omega(\log n)$ unless $P = NP$. Thus the buy-at-bulk problems are also hard to approximate within an $\Omega(\log n)$ factor.

Results: We obtain the first poly-logarithmic approximation ratios for the single-sink as well as the multi-commodity versions of the problem. Prior to our work no non-trivial results were known even for the uniform version of the problems. The precise ratios are given below.

- An $O(\log h)$ approximation and integrality gap for the single-sink problem. Unless $P = NP$ the ratio we obtain is tight up to a constant factor.
- An $O(\log^4 h)$ approximation for the multi-commodity buy-at-bulk problem.

Related Work: Network design problems are of fundamental importance in combinatorial optimization and there is a vast literature on problems and results. We refer the reader to [28] for classical results on polynomial time algorithms and to [13, 21, 30, 22, 12] for results and pointers on approximation algorithms. Here we briefly discuss the known results and techniques for some specific problems that are closely related to the problems we consider.

The known results on approximation algorithms for buy-at-bulk network design are essentially for the edge-weighted problems. For the uniform case Awerbuch and Azar [3] gave a reduction to the problem of approximating a finite metric via random tree metrics and this results in an $O(\log n)$-approximation using [10]. An improved $O(1)$-approximation is obtained for the uniform single-sink case first by Guha, Meyerson and Munagala [16] with further improvements and simplifications in [29, 17]. A special case of the multi-commodity flow version, known as the rent-or-buy problem also admits a constant factor approximation [18]. The non-uniform versions of the problem turn out to be harder. For the single-sink case, Meyerson, Munagala and Plotkin [25] obtained an $O(\log h)$-approximation. Their randomized algorithm was derandomized via an LP relaxation in [7]. For the multi-commodity problem the first non-trivial result is due to Charikar and Karagiazova [5] who obtained an $O(\log D \exp(O(\sqrt{\log h \log\log h})))$-approximation. Very recently, the authors of this paper obtained a first poly-logarithmic approximation [6]. The ratio obtained is $O(\log^4 h)$. Andrews [1] showed an $\Omega(\log^{1/4-\epsilon} n)$-hardness for the uniform case and an $\Omega(\log^{1/2-\epsilon} n)$-hardness for the non-uniform case, both under the assumption that $NP \not\subseteq ZTIME(n^{\mathrm{polylog}(n)})$. For the non-uniform single-sink case Chuzhoy et al. [9] showed an $\Omega(\log\log n)$-hardness of approximation under the assumption that $NP \not\subseteq DTIME(n^{\log\log\log n})$. We note that the inapproximability results for the edge-weighted problems apply to their node-weighted generalizations.

Techniques: The results in this paper build upon several ideas from the related work we described above. The high level framework is based on our recent work on the edge-weighted non-uniform buy-at-bulk problem [6] which reduces a multi-commodity problem to essentially a variant of its corresponding single-sink problem. The main contribution of this paper is an $O(\log h)$-approximation algorithm (and integrality gap) for the single-sink problem in the node-weighted case. The algorithm adapts ideas from several papers [24, 15, 25, 7].

Organization: Several technical details including the reduction of the buy-at-bulk problem to the two-cost network design problem are described in Section 2. In Section 3, we present the $O(\log h)$-approximation algorithm for node-weighted non-uniform single-sink buy-at-bulk problem. We present approximation algorithm for node-weighted non-uniform multi-commodity buy-at-bulk in Section 4. In Section 5 we sketch a greedy algorithm that obtains a ratio of $O(\log^3 h \log D)$ where $D = \sum_i \delta_i$ is the total demand.

2 Preliminaries

All graphs we consider are undirected. As mentioned earlier, we consider the settings in which both *edges* and *nodes* have weights. However, we can easily transform this settings into one in which only the nodes have weights by subdividing every edge (i.e. replacing it with a path of length 2) and giving the new node the weight equal to the weight of original edge. Using the same transformation, it is easy to see that the edge-weighted versions of all the problems we mentioned earlier can be reduced to their node-weighted counterparts.

It is algorithmically convenient to reduce the buy-at-bulk problem to a *two-cost* network design problem [25, 6]. This involves approximating the monotone sub-additive cost function f_v for each v by a collection of linear cost functions as follows. Let $D = \sum_i \delta_i$ be the total demand. Let $\epsilon > 0$ be any fixed constant. For $1 \leq i \leq \lceil \log D \rceil$ we define $f_v^i : \mathcal{R}^+ \to \mathcal{R}^+$ as $f_v^i(x) = f_v((1+\epsilon)^i)(1 + x/(1+\epsilon)^i)$. We replace v by a collection of nodes $S_v = \{v_i : 1 \leq i \leq \lceil \log D \rceil\}$ and the function associated with v_i is f_v^i. If uv was an edge in the original graph G we add edges $u_i v_j$ for all pairs i, j. Note that in the new instance each function is of the form $a + bx$. It can be verified that this transformation loses at most a factor of $2 + \epsilon$ in the approximation ratio. The linear functions allow us to reformulate the objective function of the buy-at-bulk network design problem. In this setting, an instance of node-weighted non-uniform multi-commodity buy-at-bulk (NMC-BB) consists of a graph G and demand pairs $\mathcal{T} = \{s_1t_1, s_2t_2, \ldots, s_ht_h\}$. Each $s_i, t_i \in V$ has a demand $\delta_i \geq 0$. We are given two separate functions $c : V \to \mathcal{R}^+$ and $\ell : V \to \mathcal{R}^+$; we call $c(v)$ and $\ell(v)$ the *cost* and *length* of v, respectively. We think of c_v as the fixed cost of v and ℓ_v as the incremental or flow-cost of v. The goal is to find a minimum cost feasible solution where a feasible solution consists of a subset of nodes $V' \subseteq V$ that includes all the terminals. The subset V' implicitly specifies the induced subgraph $G' = G[V']$. The cost of the solution specified by V' is given as

$$c(V') + \sum_{i=1}^{h} \delta_i \cdot \ell_{G'}(s_i, t_i), \tag{2.1}$$

where $c(V') = \sum_{v \in V'} c_v$ and $\ell_{G'}(u, v)$ is the shortest ℓ-node-weighted path distance between u and v in G' (the length of the end points of a path are counted as well). We obtain the single-sink problem (NSS-BB) when all the pairs share a sink (root) r. A feasible solution is a connected subgraph F containing r and spanning all the terminals. The goal is to route $\delta(t)$ units of flow from each terminal t to root r.

In the rest of the paper, we restrict our attention to the two-cost network design formulation of NMC-BB and NSS-BB.

Our algorithm for NMC-BB is a greedy iterative algorithm. In each iteration the algorithm finds a partial solution (a solution that connects some of the remaining pairs) at low density, where the density is the ratio of the cost of the partial solution to the number of new pairs it connects. We will use the following basic lemma in the analysis of these algorithms (see e.g., [23]).

LEMMA 2.1. *Suppose that an algorithm works in iterations and in iteration i it finds a partial solution $V_i \subseteq V$ that routes a new subset T_i of the demands. Let* OPT *be the cost of an optimum solution and u_i be the number of unrouted demands at the time V_i is found. If for every i, the cost of the partial solution $G[V_i]$ over the number of pairs it routes is at most $f(h) \cdot \frac{\text{OPT}}{u_i}$, then the cost of the solution returned by the algorithm is at most $f(h) \cdot (\ln h + 1) \cdot$ OPT.*

3 The Single-Sink Problem

Recall that the instance to NSS-BB is an undirected graph $G = (V, E)$ with a designated root node r, a set of terminals $T \subseteq V$, a demand function $\delta : T \cup \{r\} \to \mathbb{R}^+$, a cost function $c : V \to \mathbb{R}^+$, and a length function $\ell : V \to \mathbb{R}^+$. Our main result of this section is:

THEOREM 3.1. *There is a deterministic $O(\log h)$-approximation algorithm for NSS-BB where h is the number of terminals. Furthermore, the integrality gap of a natural linear programming relaxation for the problem is $O(\log h)$.*

We assume without loss of generality that $c(r) = \ell(r) = 0$; we can arrange this by adding a dummy root to the original root. Thus we may assume that $\delta(r)$ is large enough (technically ∞); this will subsequently help simplify the description of our algorithm.

Since NSS-BB generalizes the node-weighted Steiner tree, which has a $\Omega(\log n)$-hardness [24] (via a simple reduction from set-cover):

COROLLARY 3.1. *NSS-BB has an approximability threshold of $\Theta(\log n)$, unless* P = NP.

The algorithm for Theorem 3.1 uses ideas from the works of Klein and Ravi [24], Guha et al. [15], Meyerson et al. [25] and Chekuri et al. [7]. In particular, we use the spider ideas from [24] and randomized merging from [25].

A *spider* is a tree with at most one node of degree more than two and such that all its leaves are terminals. Furthermore no internal node is a terminal. The *center* of a spider is a node from which there are edge-disjoint paths to the leaves of the spider. So if the spider has a node of degree at least three, its center is unique. The density of a spider is the ratio of its total cost to the number of terminals in it where the total cost depends on the problem definition. For the node-weighted Steiner tree problem the total cost is just the sum of the weights of the nodes in the spider. For this problem, Klein and Ravi [24] showed the existence of a decomposition of the optimum solution into spiders. Therefore, there is always a spider whose density is no more than the density of the optimum (which is the ratio of the cost of the optimum to the total number of terminals). They also show how to find a minimum density spider in polynomial time. Given this tool in hand, they iteratively find a minimum density spider and contract the spider into a single node, until all the terminals are contracted into r. Again, using a standard set-cover type analysis, this yields an $O(\log n)$-approximation for the node-weighted Steiner tree problem. Guha et al. [15] later showed that in fact the density of the minimum density spider is no more than the density of an optimum solution to the natural linear programming relaxation to the problem.

A randomized algorithm for NSS-BB: We describe a randomized algorithm for NSS-BB that is inspired by the spider approach of [24] and the randomized merging algorithm of [25] for the edge-weighted NSS-BB. To describe the algorithm we first define the cost of a spider in the setting of NSS-BB. Here we restrict our attention to spiders for which the center is prescribed. For a spider $\mathcal{S}$ we let $T(\mathcal{S})$ be the terminals at the leaves of the spider. The total cost of a spider $\mathcal{S}$ with center s is:

$$c(s) + \sum_{t\in T(\mathcal{S})} (c(p_t) - c(s) + \delta(t)\cdot \ell(p_t)),$$

where p_t is the path between t and s in $\mathcal{S}$ and $c(p_t)$ and $\ell(p_t)$ being the sum of the costs and lengths of the nodes on this path, respectively. Although the definition of a spider requires the paths p_t, $t \in \mathcal{S}$ to be internally node-disjoint, we abuse notation and allow the paths to share nodes. However in this case the cost of the spider would count the cost of a shared node multiple times, one for each of the paths containing it, as defined above. The randomized algorithm RandSpider is described in Fig 1. We make two observations. If the root is a terminal in the minimum density spider then by our technical assumption that $\delta(r) = \infty$ the root will be chosen as the proxy. In the last step, it is not necessary for a non-proxy terminal to connect to the proxy terminal using the path in $\mathcal{S}$ - there could be a cheaper direct path, however the analysis carries through using the path in $\mathcal{S}$. We can prove, using ideas similar to those in [25] that RandSpider yields a solution of expected cost $O(\log h)$OPT where OPT is the value of an optimum integral solution. We prove a stronger theorem which also yields a bound on the integrality gap of a natural linear programming relaxation. First we show that a minimum density spider can be computed in polynomial time.

For a terminal t and a node v we let $d_t(v)$ denote $\min_{p\in P_{tv}}(c(p)+\delta(t)\ell(p))$ where P_{tv} is the set of all paths between t and v. In other words $d_t(v)$ is the shortest path distance between t and v with the weight of a node u set to $c(u) + \delta(t)\ell(u)$.

Lemma 3.1. *Given an instance of NSS-BB we can find a minimum density spider in polynomial time.*

Proof. By enumerating over all nodes, we can assume that we have correctly guessed the center s of the minimum density spider in the given instance. For simplicity, we assume that s is not a terminal - by hanging dummy terminals this can always be ensured. Without loss of generality assume that terminals are ordered such that $d_{t_1}(s) \le d_{t_2}(s) \le \ldots \le d_{t_h}(s)$. Let P_i be a path from t_i to s that certifies $d_{t_i}(s)$. For $2 \le j \le h$, let $\alpha_j = \frac{1}{j}\cdot(c(s) + \sum_{1\le i\le j}(d_{t_i}(s) - c(s)))$ denote the density of a subgraph obtained by connecting the first j terminals (in the ordering) to s. Let $j^* = \mathrm{argmin}_j \alpha_j$. We return the subgraph $\mathcal{S}$ obtained by the union of the paths $P_1, P_2, \ldots, P_{j^*}$. It is easy to see that the density of $\mathcal{S}$ is no more than the density of a minimum density spider. □

A linear programming relaxation for NSS-BB: We first formulate NSS-BB as an IP for which we have the following LP relaxation. For $t \in T$, let $\mathcal{P}_t$ denotes the set of directed paths from root r to t. We assume that the terminals are at distinct nodes and hence $\mathcal{P}_t \cap \mathcal{P}_{t'} = \emptyset$ for $t \ne t'$. For $v \in V$, a variable $x(v) \in [0,1]$ indicates whether v is chosen in the solution or not. For $p \in \cup_t \mathcal{P}_t$ a variable $f(p) \in [0,1]$ indicates whether p is used to connect a terminal to the root. We use $\ell(p)$ to denote $\sum_{v\in p}\ell(v)$. The LP assigns fractional capacities to nodes such that one unit of flow can be shipped from each terminal t to the root.

LP-NSS:

$$\min \quad \sum_{v\in V} c(v)\cdot x(v) + \sum_{t\in T}\delta(t)\sum_{p\in\mathcal{P}_t}\ell(p)\cdot f(p)$$

$$\begin{array}{rcll} \sum_{p\in P_t | v\in p} f(p) & \le & x(v) & v\in V,\ \ t\in T \\ \sum_{p\in\mathcal{P}_t} f(p) & \ge & 1 & t\in T \\ x(v), f(p) & \ge & 0 & v\in V,\ p\in\cup_t\mathcal{P}_t \end{array}$$

Let OPT_{LP} be the value of an optimum solution to LP-NSS. We prove that RandSpider yields an integral solution of expected cost $O(\log h)\text{OPT}_{LP}$. The key technical lemma for this is the following.

Lemma 3.2. *For any instance of NSS-BB there is a spider of density at most* OPT_{LP}/h.

Algorithm RandSpider for NSS-BB:

1. If root is the only terminal return the tree $\{r\}$.
2. Compute a minimum density spider $\mathcal{S}$.
3. Choose a *proxy* terminal t from $T(\mathcal{S})$ such that probability of $t' \in T(\mathcal{S})$ being chosen is exactly $\delta(t')/\delta(T(\mathcal{S}))$. Set the demand of t to be equal to $\delta(T(\mathcal{S}))$ and remove terminals in $T(\mathcal{S}) - \{t\}$.
4. Recursively obtain a solution to the reduced problem.
5. Connect each non-proxy terminal in $T(\mathcal{S})$ to the root via t using the path in $\mathcal{S}$.

Figure 1: A randomized algorithm for NSS-BB

We assume the lemma and prove Theorem 3.1. Although RandSpider is a randomized algorithm we will be able to obtain a deterministic algorithm using the linear programming relaxation using ideas similar to [7].

Let I be the given instance and let $\mathcal{S}$ be a minimum density spider for I computed by RandSpider in Step 2. Let I' be the reduced instance obtained after the proxy terminal from $\mathcal{S}$ is chosen in Step 3 of the algorithm. Let $\text{OPT}_{LP}(I)$ and $\text{OPT}_{LP}(I')$ denote the optimum values of LP-NSS on I and I' respectively. Note that $\text{OPT}_{LP}(I')$ is a random variable.

Lemma 3.3. $\mathbf{E}[\text{OPT}_{LP}(I')] \le \text{OPT}_{LP}(I)$.

Proof. Let x^*, f^* be an optimal feasible solution to the instance I. In the instance I' we have essentially changed only the value of the demands; the proxy terminal gets a demand equal to $\delta(T(S))$ while the removed terminals get demand 0. Thus the solution x^*, f^* is also a feasible solution to I'. We show that the expected cost of this solution for I' is the same as $\text{OPT}_{LP}(I)$. For terminal $t \in T_i$ let $\alpha(t) = \sum_{p\in\mathcal{P}_t} \ell(p) \cdot f^*(p)$. We have $\text{OPT}_{LP}(I) = \sum_{v\in V} c(v) \cdot x^*(v) + \sum_{t\in T} \delta(t) \cdot \alpha(t)$. For every terminal $t \notin T(\mathcal{S})$, its contribution to the total cost remains unchanged in the solution for I'. On the other hand, the expected contribution of a terminal $t \in T(\mathcal{S})$ in I' is exactly $\delta(t)\alpha(t)$ for the following reason; the probability that t is chosen as a proxy terminal is $\delta(t)/\delta(T(S))$ and if it is chosen then the contribution is $\delta(T(S))\alpha(t)$. Thus it can be seen that the expected cost of the solution x^*, f^* for I' is at most $\text{OPT}_{LP}(I)$. □

For a spider $\mathcal{S}$ let $\beta(\mathcal{S})$ denote its cost as defined earlier.

Lemma 3.4. *In Step 5 of RandSpider, the expected cost of routing non-proxy terminals to the chosen proxy terminal is at most* $2\beta(\mathcal{S})$.

Proof. We can bound the expected cost as follows. The cost consists of two parts. The first part accounts for the cost of each terminal $t \in T(S)$ sending its demand to the center s of $\mathcal{S}$. This cost is deterministically equal to $\beta(\mathcal{S})$, by definition. The second part accounts for the center sending the total demand $\delta(T(\mathcal{S}))$ to the chosen proxy terminal. The expected cost of this second part is seen to be $\sum_{t\in T(\mathcal{S})} a_t \delta(T(\mathcal{S}))\ell(p_t)$ where a_t is the probability that t is chosen as the proxy terminal and p_t is the path from t to the center s in $\mathcal{S}$. Since $a_t = \delta(t)/\delta(T(\mathcal{S}))$ it follows that the expected cost is $\sum_{t\in T(\mathcal{S})} \delta(t)\ell(p_t)$ which is at most $\beta(\mathcal{S})$. Therefore the total expected cost is at most $2\beta(\mathcal{S})$. □

Proof of Theorem 3.1. We first prove that RandSpider yields a solution of *expected* cost at most $3H_h\text{OPT}_{LP}$ where $H_h = 1 + 1/2 + \ldots + 1/h$ is the h'th Harmonic number. This immediately proves that the integrality gap of LP-NSS is $O(\log h)$. We then sketch a way to derandomize RandSpider using pessimistic estimators.

Let I be the given instance of NSS-BB. If $h = 1$ then it can be easily checked that the algorithm returns an optimum solution. Consider the steps of RandSpider on I. Let $\mathcal{S}$ be the spider computed in Step 2 and let k be the number of terminals in $\mathcal{S}$. By Lemma 3.2, we have that $\beta(\mathcal{S})/k \le \text{OPT}_{LP}(I)/h$. Let I' be the random problem that RandSpider generates in Step 3. By induction the expected cost of the solution produced by RandSpider to I' is at most $3H_{h'}\text{OPT}_{LP}(I')$ where $h' = h - k + 1$ is the number of terminals in I'. The total cost of the solution for I is bounded by the cost of the solution to I' and the cost of the routing of non-proxy terminals in $\mathcal{S}$ to the chosen proxy terminal. By linearity of expectation and using Lemma 3.3 and Lemma 3.4, the expected cost of the solution to I is at most

$$\begin{aligned} 3H_{h'}\text{OPT}_{LP}(I) + 2\beta(\mathcal{S}) &\le (3H_{h'} + 2k/h)\text{OPT}_{LP}(I) \\ &\le 3H_h\text{OPT}_{LP}(I). \end{aligned}$$

The algorithm RandSpider can be derandomized using a solution to LP-NSS using the same ideas as in [7]. Let x^*, f^* be a feasible solution to LP-NSS on I. In Step 3 of the algorithm, instead of choosing the proxy terminal in $\mathcal{S}$ at random we can pick the terminal deterministically as follows. For $t \in T(\mathcal{S})$ let I'_t be the instance obtained if t is chosen as a proxy terminal. And let β_t be the cost of routing the terminals in $T(\mathcal{S}) - \{t\}$ to t using $\mathcal{S}$ and let α_t be the value of the solution x^*, f^* on I'_t. The above analysis shows that there exist a terminal $t' \in T(\mathcal{S})$ such that $3H_{h'}\alpha_t + 2\beta_t \leq 3H_h\text{OPT}_{LP}(I)$. Note that α_t and β_t can be computed in polynomial time from x^*, f^* and $\mathcal{S}$ and thus we can identify t' by evaluating $3H_{h'}\alpha_t + 2\beta_t$ for each $t \in T(\mathcal{S})$. We deterministically choose t' to be the proxy terminal for $\mathcal{S}$, solve the problem I'_t recursively and connect the terminals in $T(\mathcal{S}) - \{t'\}$ to the root via t' using $\mathcal{S}$. Inductively the cost of the solution on I'_t is at most $3H_{h'}\alpha_t$. Therefor the total cost of the solution is $3H_{h'}\alpha_t + 2\beta_t \leq 3H_h\text{OPT}_{LP}(I)$ as desired. □

The remainder of this section is devoted to the proof of Lemma 3.2. The proof is similar in spirit to that of Guha et al. [15] for the node-weighted Steiner tree problem. Let R be the density of a minimum density spider for the given instance. We wish to show that $R \leq \text{OPT}_{LP}/h$ or in other words $\text{OPT}_{LP} \geq hR$. We prove this by exhibiting a feasible solution to the dual of LP-NSS which is given below.

DP-NSS:

$$\max \quad \sum_{t \in T} y(t)$$

$$\begin{array}{rcll} \sum_{t\in T} z_t(v) & \leq & c(v) & v \in V \\ y(t) - \sum_{v \in p} z_t(v) & \leq & \delta(t) \cdot \ell(p) & p \in \mathcal{P}_t, \quad t \in T \\ y(t), z_t(v) & \geq & 0 & v \in V, t \in T \end{array}$$

We create a solution y', z' to DP-NSS as follows. Recall that $d_t(v) = \min_{p \in P_{tv}} c(p) + \delta(t)\ell(p)$. We set $y'(t) = R$ for each terminal t. We set $z'_t(v) = \max\{0, \min\{c(v), R - d_t(v) + c(v)\}\}$ for each terminal t and node v. Note that $0 \leq z'_t(v) \leq c(v)$ for each t, v. If y', z' is a feasible solution to DP-NSS then by weak duality $\text{OPT}_{LP} \geq \sum_t y'(t) = hR$; this is the desired inequality.

We claim a simple property of z'.

Claim 3.5. *Let $\gamma_t(v) = \min_{p \in P_{tv}} z'_t(p) + \delta(t)\ell(p)$. Then $\min\{R, d_t(v)\} \leq \gamma_t(v) \leq d_t(v)$.*

The above claim can be shown by an induction on the length of the path defining $\gamma_t(v)$ but the formal details are cumbersome and we omit them.

Lemma 3.6. *The solution y', z' is feasible for DP-NSS.*

Proof. We claim that $d_t(r) \geq R$ for each t. If not, the spider obtained by connecting t to r would have density $d_t(r) < R$. Therefore by Claim 3.5, we have $y'(t) = R \geq \gamma_t(r)$ which implies that for each $p \in \mathcal{P}_t$,

$$y'(t) \leq \sum_{v \in p} z'_t(v) + \delta(t)\ell(p).$$

Now consider any node v. We claim that

$$\sum_t z'_t(v) \leq c(v).$$

Suppose the above fails for a node s, that is $\sum_t z'_t(s) > c(s)$. Let $T_s = \{t \mid z'_t(s) > 0\}$. Note that $|T_s| \geq 2$ since $z'_t(v) \leq c(v)$ for all t, v. We will prove that the spider $\mathcal{S}$ with center s and terminals T_s has density strictly less than R, a contradiction. The cost of the spider $\mathcal{S}$ is

$$\begin{aligned} c(s) + \sum_{t \in T_s} (d_t(s) - c(s)) &< \sum_{t \in T_s} z'_t(s) + \sum_{t \in T_s} (d_t(s) - c(s)) \\ &= \sum_{t \in T_s} (d_t(s) - c(s) + z'_t(s)) \\ &\leq \sum_{t \in T_s} R \\ &\leq |T_s| R. \end{aligned}$$

Therefore the density of $\mathcal{S}$ is strictly less than R. The penultimate inequality above follows from the definition of $z'_t(s)$ and the fact that $z'_t(s) > 0$ for each $t \in T_s$. □

4 The Multi-Commodity Problem

In this section we prove the following.

Theorem 4.1. *There is a polynomial time algorithm for NMC-BB with an $O(\log^4 h)$ approximation ratio, where h is the number of pairs.*

The general structure of the algorithm is similar to that in [6] for the MC-BB and follows an iterative greedy scheme. In each iteration we find a partial solution that connects a subset of the pairs that remain at the beginning of the iteration. The connected pairs are then removed. The *density* of the partial solution is the ratio of the total cost of the partial solution to the number of pairs in the solution. We prove that the density of the partial solution computed at every iteration is a polylogarithmic factor away from the density of the optimum solution. As in [6], a key ingredient in our proof is to show the *existence* of a partial solution with a very restricted structure, called a *junction-tree*. Given a subset A of the pairs, a junction tree for A rooted at r is a tree T containing the end points of all pairs in A such that the unique path connecting every pair of A goes via r. The cost of the junction-tree T is

$$\sum_{v \in V(T)} c_v + \sum_{s_i t_i \in A} \delta_i \cdot (\ell_T(r, s_i) + \ell_T(r, t_i)).$$

In other words, the pairs in A connect via the junction r. Note that if the set A and r are known, a junction-tree is essentially an instance of the single-sink problem NSS-BB. We prove that given an instance of NMC-BB there is always a low density partial solution that is a junction-tree. The problem of finding a low density junction-tree is closely related to the density variation of NSS-BB, called den-NSS-BB in which we want to find a solution with minimum density i.e. the ratio of total cost over the number of terminals spanned (v.s. the total cost as in SS-BB). We use Theorem 3.1 and obtain an $O(\log^2 h)$-approximation for den-NSS-BB and by a slight modification a similar ratio for finding a minimum density junction-tree. Putting together these ingredients give us the poly-logarithmic approximation for NMC-BB.

4.1 A Junction Tree Lemma We prove the following lemma on the existence of a junction tree with low density.

LEMMA 4.1. *Given an instance of MC-BB on h pairs there exists a junction-tree of density $O(\log h) \cdot \frac{\text{OPT}}{h}$.*

The rest of this subsection is devoted to the proof of the above lemma. The proof for the node-weighted case essentially follows the proof for the edge-weighted case [6]. In [6] proofs are given for two lemmas with slightly weaker bounds and a proof for a bound of $O(\log h) \cdot \frac{\text{OPT}}{h}$ was claimed (the idea for this stronger bound was suggested by Harald Räcke). Here we give the details of the proof as adapted to the node-weighted setting. We need the following technical lemma.

LEMMA 4.2. ([6]) *Given an instance of NMC-BB on $G = (V, E)$ there is an optimum solution $G = G[V^*]$ such that the number of nodes in G^* of degree more than 2 is at most* $\min(n, h^2)$.

Given an instance of MC-BB on a graph $G = (V, E)$, let $V^* \subseteq V$ induce an optimum solution for the given instance. Using Lemma 4.2, we can assume that $G[V^*]$ has $O(\min(n, h^2))$ nodes by suppressing non-terminals that have degree at most 2 in G^*. Recall that the optimum solution value, OPT is $c(V^*) + \sum_i \delta_i \ell_{G^*}(s_i, t_i)$ where $\ell_{G^*}(s_i, t_i)$ is the ℓ-node-weighted distance between s_i and t_i in G^*.

The crucial ingredient in the proof is the existence of a hierarchical decomposition of an undirected *edge*-weighted graph that has certain useful properties to be described below. Our focus is on node-weights and we have two weight functions c and ℓ. In the following we use ℓ to define the edge-weights of G^* by setting for each edge $uv \in E(G^*)$ a weight $\ell(uv) = \ell(u) + \ell(v)$. Note that for any $x, y \in V(G^*)$ the distance in G^* with ℓ-edge-weights is within a factor of 2 of the distance with ℓ-node-weights. The hierarchical decomposition of this edge-weighted graph will be used later. We think of the decomposition as induced by a laminar family of subsets of nodes of the graph; it is convenient to represent the laminar family by a rooted tree with the leaves of the tree corresponding to the nodes of the graph. Although the proof of the required laminar family essentially follows from Bartal's first construction of metric embeddings of graphs into trees [4], we keep the discussion somewhat abstract to isolate the desired properties.

Given an edge-weighted graph $G = (V, E)$ let $T = (V_T, E_T)$ be a tree representing a laminar family on V. We let $d_G(a, b)$ denote the distance in G between nodes a and b where the distance is defined with respect to the given edge-weights. For an internal node $u \in V_T$ let T_u be the subtree of T rooted at u. We denote by G_u the subgraph of G induced by the leaves in T_u. For a pair of nodes $a, b \in V(G)$, let $G^T_{a,b}$ denote the graph G_u where u is the least common ancestor of a and b in T. We denote by $\Delta_T(a, b)$ the diameter of the graph $G^T_{a,b}$. Note that, trivially, $\Delta_T(a, b) \geq d_G(a, b)$ where $d_G(a, b)$ is the distance between a, b in G.

Given G and a laminar family T we say that a pair of nodes $a, b \in V(G)$ is α-good in T iff $\Delta_T(a, b) \leq \alpha \cdot d_G(a, b)$.

LEMMA 4.3. *Given an n-node edge weighted graph $G = (V, E)$, there is a probability distribution on laminar families on G such that for a tree T picked from the distribution, the following is true: there exists a universal constant c such that for any pair $a, b \in V(G)$*

$$\Pr[\Delta_T(a, b) \leq c \log n \cdot d_G(a, b)] \geq 1/2.$$

Proof. In [4], Bartal created a distribution of laminar families that yields a probabilistic embedding of a graph metric into trees with $O(\log^2 n)$ distortion. The rest of the argument below shows that the same distribution satisfies the properties that we desire.

We briefly sketch the construction in [4]. Given a graph G, a procedure is given that randomly partitions $V(G)$ into $V_1, V_2, \ldots, V_k$ such that the following two properties hold: (i) for $1 \leq i \leq k$, the diameter of $G_i = G[V_i]$ (also known as the *strong* diameter) is at most $\Delta(G)/2$ and (ii) there is a universal constant $c' > 0$ such that for every pair of nodes a, b, the probability that a, b are in different parts is at most $c \log n \cdot d_G(a, b)/\Delta$. The laminar family for G is obtained by applying the partitioning procedure recursively to the graphs $G_1, G_2, \ldots, G_p$. Let T be the random laminar family produced by the process.

Consider a pair of nodes $a, b \in V(G)$. We observe that $\Delta_T(a, b)$ is the diameter of the smallest graph in the family with both a, b in the graph. We estimate the probability, p, that this diameter is larger than $c \log n \cdot d_G(a, b)$. For simplicity, we assume that the diameter of the graphs decreases exactly by a factor of 2 as the recursion proceeds - this assumption can be easily dispensed with. Let p_i be the probability that a, b are separated at level i of the recursion conditioned on the fact that they are not separated in levels 1 to $i-1$. From the random partitioning procedure, $p_i \leq c' \log n \cdot 2^{i-1} d_G(a, b)/\Delta$. We can therefore upper bound p by $p_1 + p_2 + \ldots + p_h$ where h is the largest integer such that $\Delta/2^h \leq c \log n \cdot d_G(a, b)$. It can be seen that $p \leq 1/2$ for $c \geq 4c'$. $\square$

Corollary 4.1. *Given G and set of pairs of nodes A, there exists a laminar family T such that the number of pairs in A that are $2c \log n$-good in T is at least $|A|/4$.*

Now we prove the junction tree lemma.

Proof of Lemma 4.1. We assume without loss of generality that G^* is connected, otherwise we can work with each connected component separately. We convert the ℓ-node-weights into ℓ-edge-weights in G^* as described earlier. We apply Corollary 4.1 to the edge-weighted graph G^* and the set of input pairs $\mathcal{T}$ to obtain a tree T. Let $\mathcal{T}'$ be the pairs that are $O(\log h)$-good in T. We do a path-decomposition of T as follows. We obtain the first path P_1 by walking from the root down to a leaf where, at each step, the walk chooses a child of the current node that has the largest number of leaves in its subtree. We then remove P_1 from T and apply the same procedure recursively to each of the trees in $T \setminus P_1$. Let $P_1, P_2, \ldots, P_k$ be the non-singleton paths obtained from the procedure. Let r_i and u_i be the internal node and the leaf end points of P_i. Let $H_i = G^*_{r_i}$. We call each H_i a cluster and we call u_i its center. We observe that the paths are node disjoint. We make another useful claim: any node $u \in V(G^*)$ is in $O(\log h)$ clusters. This claim follows from the choice of the heaviest child in the walk to create each path.

We create a junction tree T_i in each cluster H_i as follows. For a pair $(a, b) \in \mathcal{T}'$ we place it in the tree T_i iff the least common ancestor of a and b belongs to $V(P_i)$. We set the root of T_i to be u_i. We claim that one of these junction trees has density $O(\log h)\frac{\text{OPT}}{h}$. For this purpose we compute the total fixed cost and the total incremental cost of all the junction trees. Consider the tree T_i and a pair (a, b) from $\mathcal{T}'$ in T_i. Since the pair (a, b) is $O(\log h)$-good, it follows that $d_{H_i}(a, u_i) = O(\log h) d_{G^*}(a, b)$ and $d_{H_i}(b, u_i) = O(\log h) d_{G^*}(a, b)$. Recall that distances in the edge-weighted graph G^* approximate the ℓ-node-weighted distances within a factor of 2. Therefore the total incremental cost over all junction trees is $O(\log h)\text{OPT}_\ell$. Recall that every node $u \in V(G^*)$ is in $O(\log h)$ clusters. Therefore the total fixed cost of all the junction trees is $O(\log h)\text{OPT}_c$. Thus the total cost of all junction trees is $O(\log h)\text{OPT}$. We also have that $|\mathcal{T}'| \geq |\mathcal{T}|/4$. This finishes the proof of the claim, and hence the lemma. $\square$

4.2 Finding an approximate min-density junction tree

In this subsection we give an $O(\log^2 h)$-approximation algorithm for den-NSS-BB and min-density junction tree. The algorithm and analysis are built upon the LP relaxation and the proof of the integrality gap for NSS-BB shown in Section 3. We restrict our attention to the rooted version where the goal is to find a minimum density junction tree rooted at a given root node r. The unrooted problem can be reduced to the rooted problem by trying each node as the root and picking the best of the solutions. Consider the following LP relaxation of den-NSS-BB which modifies LP-NSS. For each terminal t_i, we have an additional variable y_i that indicates whether t_i is chosen in the solution or note. We normalize $\sum_t y_t$ to 1.

LP-NSSD:

$$\min \quad \sum_{v \in V} c(v) \cdot x(v) + \sum_{t \in T} \delta(t) \sum_{p \in \mathcal{P}_t} \ell(p) \cdot f(p)$$

$$\begin{array}{rcll} \sum_{t \in T} y_t & = & 1 & \\ \sum_{p \in P_t | v \in p} f(p) & \leq & x(v) & v \in V, \ \ t \in T \\ \sum_{p \in \mathcal{P}_t} f(p) & \geq & y_t & t \in T \\ x(v), f(p), y_t & \geq & 0 & v \in V, \ p \in \cup_t \mathcal{P}_t \end{array}$$

Theorem 4.2. *There is an $O(\log^2 h)$-approximation for den-NSS-BB.*

Proof. The proof is similar to that of Theorem 4.2 in [6]. The main difference is that here we use Theorem 3.1. Consider an optimum solution to LP-NSSD. We obtain disjoint subsets of the terminals $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_p$ as follows. Let $y_{\max} = \max_t y_t$. For $0 \leq a \leq 2\lceil \log h \rceil$, let $\mathcal{T}_a = \{t_j \mid y_{\max}/2^{a+1} < y_{t_j} \leq y_{\max}/2^a\}$. Thus $p = 1 + 2\lceil \log h \rceil = O(\log h)$. It is easy to see that there is an index b such that $\sum_{t_j \in \mathcal{T}_b} y_{t_j} = \Omega(1/\log h)$. From this we also have that $2^b/|\mathcal{T}_b| = O(\log h)$. We now solve an NSS-BB instance on $\mathcal{T}_b$. We claim that the resulting solution is an $O(\log^2 h)$-approximation to den-NSS-BB. To prove this, let α be the value of the optimum solution to LP-NSSD on the given instance. Note that if we scale up, by a factor of $2^{b+1}/y_{\max}$, the given optimum solution to LP-NSSD we obtain a feasible solution to LP-NSS on the terminal set $\mathcal{T}_b$. The cost of this scaled solution to LP-NSS is $2^{b+1}\alpha$. Since the integrality gap of LP-NSS is $O(\log h)$ (by Theorem 3.1), we obtain an integral solution that connects each terminal in $\mathcal{T}_b$ to the root such that cost of the solution is $O(\log h) \cdot 2^{b+1}\alpha$. The density of this solution is therefore $O(\log h) \cdot 2^{b+1}\alpha/|\mathcal{T}_b|$ which is $O(\log^2 h)\alpha$. Thus the integrality gap of LP-NSSD is $O(\log^2 h)$ yielding the desired approximation. $\square$

COROLLARY 4.2. *There is an $O(\log^2 h)$-approximation for computing min-density junction tree.*

Proof. Given an instance of NMC-BB, we consider each source or sink as a terminal. Also, for every pair s_i, t_i we add the following set of constraints to the LP-NSSD: $y_{s_i} = y_{t_i}$. This ensures that either we include both of s_i and t_i in the tree or none of them. The rounding scheme in the proof of Theorem 4.2 extends to this LP and so we get an $O(\log^2 h)$-approximation for the min-density junction tree problem. $\square$

5 A Greedy Approximation Algorithm

Here we describe the overview of a greedy algorithm for NMC-BB with ratio $O(\log^3 h \cdot \log D)$. When D is polynomial in h the performance of this algorithm is asymptotically the same as the algorithm described in Section 4. The greedy algorithm is essentially the same as the greedy algorithm for MC-BB in [6]. It finds a partial solution with good density at every iteration. We describe briefly the general idea of the algorithm for MC-BB (from [6]) and the differences with the one for NMC-BB. One of the ingredients for the algorithm is the existence of a junction tree with some additional properties; the proof of existence of such trees for the edge-weighted case, as shown in [6] (see Lemma 3.4 and Claim 3.5), can be extended to the node-weighted case in a straight forward fashion. The proof applies to the unit-demand case ($\delta_i = 1$ for each i) and this results in the approximation ratio depending on $\log D$. The main ingredient in the greedy algorithm for MC-BB is an approximation algorithm for the shallow-light trees described below

Shallow-light k-Steiner Tree (KSLT): The instance to shallow-light k-Steiner problem is a graph $G(V, E)$, with edge-weight function $c : E \to \mathcal{R}^+$ and edge-length function $\ell : E \to \mathcal{R}^+$, a collection T of terminals containing a root s, a number k, and a diameter bound L. The goal is to find a minimum c-cost s-rooted k-Steiner tree (at tree that contains k terminals) that has ℓ-diameter at most L. A (ρ_1, ρ_2) bi-criteria approximation algorithm for the shallow-light k-Steiner problem finds an s-rooted k-Steiner tree with diameter at most $\rho_1 \cdot L$ and cost at most $\rho_2 \cdot B$ with B being the optimum cost for a k-Steiner tree of diameter L. The algorithm in [6], uses the following result from [20] for the *edge-weighted* version of shallow-light trees:

THEOREM 5.1. *[20] There is an $(O(\log h), O(\log^3 h))$-approximation algorithm for the edge-weighted shallow-light k-Steiner tree problem which finds a $k/8$-Steiner tree.*

The algorithm for MC-BB has a main procedure which tries to find a good density partial solution. It has a sources-phase and a sinks-phase in each phase it uses as a sub-routine the algorithm guaranteed by Theorem 5.1 to find a minimum density tree rooted at a node s. We omit the details of the analysis from [6].

Our greedy algorithm for the NMC-BB follows the same paradigm. For that we need a node-weighted version of Theorem 5.1. We define the node-weighted shallow-light Steiner trees similarly:

Node-weighted shallow-light k-Steiner tree (NKSLT): we have a graph $G(V, E)$, with node cost function $c : V \to \mathcal{R}^+$ and length function $\ell : V \to \mathcal{R}^+$, a collection T of terminals containing a root s, a number k, and a diameter bound L. The goal is to find a minimum c-node-cost s-rooted k-Steiner tree that has ℓ-node-diameter at most L.

LEMMA 5.1. *There is polynomial-time algorithm A such that, given an instance of NKSLT, finds a $k/8$-Steiner tree with ℓ-diameter at most $O(\log h \cdot L)$ and c-cost at most $O(\log^3 h \cdot \text{OPT})$ where OPT is the minimum c-cost k-Steiner tree with ℓ-diameter bound L.*

Using the above lemma, an algorithm similar to the one for MC-BB gives an $O(\log^3 h \log D)$-approximation for NMC-BB. We briefly sketch the the ideas for the proof of Lemma 5.1. The algorithm borrows ideas from the algorithm of [20] for (edge-weighted) shallow-light k-Steiner trees (Theorem 5.1) and [24] for node-weighted Steiner tree. Here we describe the similarities and differences. The algorithm for Theorem 5.1 is a greedy algorithm that starts from every terminal as a single-component. At every iteration it tries to connect two components by a "cheap" path. Once a path is found the two components are merged into one. We continue until we have a component with at least $k/8$ terminals. The informal definition of a cheap path is that we can charge the cost of the path to the nodes in the two merged components such that the cost is at most a poly-logarithmic factor of the optimum density (there are some technical details that we omit here). The algorithm for Lemma 5.1 has a similar structure. The main difference is that at each iteration, instead of finding a cheap path that connects two terminals (at good density) we find a cheap spider that potentially connects multiple components at once. The algorithm for finding a low density spider is similar to the one used in Lemma 3.1. After finding a low density spider (compared to the density of the optimum) we merge the components it spans. We continue this until there are at least $k/8$ terminals in one component. Further details and the formal proof will appear in later versions of this paper.

References

[1] M. Andrews. Hardness of buy-at-bulk network design. *Proc. of IEEE FOCS*, 115–124, 2004.

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

[3] B. Awerbuch and Y. Azar. Buy-at-bulk network design. *Proc. of IEEE FOCS*, 542–547, 1997.

[4] Y. Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. *Proc. of IEEE FOCS*, 184–193, 1996.

[5] M. Charikar and A. Karagiozova. On non-uniform multicommodity buy-at-bulk network design. *Proc. of ACM STOC*, 176–182, 2005.

[6] C. Chekuri, M. Hajiaghayi, G. Kortsarz, and M. Salavatipour. Approximation algorithms for non-uniform buy-at-bulk network design problems. *Proc. of IEEE FOCS*, 2006.

[7] C. Chekuri, S. Khanna, and J. Naor. A deterministic algorithm for the cost-distance problem. *Proc. of ACM-SIAM SODA*, 232–233, 2001.

[8] C. Chekuri, S. Khanna, and F. Bruce Shepherd. Multicommodity flow, well-linked terminals, and routing problems. *Proc. of ACM STOC*, 183–192, 2005.

[9] J. Chuzhoy, A. Gupta, J. Naor, and A. Sinha. On the approximability of some network design problems. *Proc. of ACM-SIAM SODA*, 943–951, 2005.

[10] J. Fakcharoenphol, S. Rao, and K. Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *JCSS*, 69:485–497, 2004. Preliminary version in *Proc. of ACM STOC*, 2003.

[11] U. Feige, M.T. Hajiaghayi, and J. Lee. Improved approximation algorithms for minimum-weight vertex separators. *Proc. of ACM STOC*, 563–572, 2005.

[12] L Fleischer, K. Jain, and D. P. Williamson. An iterative rounding 2-approximation algorithm for the element connectivity problem. *JCSS*, 72(5):838–867, 2006.

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

[14] S. Guha and S. Khuller. Improved Methods for Approximating Node Weighted Steiner Trees and Connected Dominating Sets. *Information and Computation*, 150(1):57–74, 1999.

[15] S. Guha, A. Moss, J. Naor, and B. Schieber. Efficient recovery from power outage. *Proc. of ACM STOC*, pages 574–582, 1999.

[16] S. Guha, A. Meyerson, and K. Munagala. A constant factor approximation for the single sink edge installation problems. *Proc. of ACM STOC*, 383–388, 2001.

[17] A. Gupta, A. Kumar, and T. Roughgarden. Simpler and better approximation algorithms for network design. *Proc. of ACM STOC*, 365–372, 2003.

[18] A. Gupta, A. Kumar, M. Pal, and T. Roughgarden. Approximation via cost-sharing: a simple approximation algorithm for the multicommodity rent-or-buy problem. *Proc. of IEEE FOCS*, 606–615, 2003.

[19] M.T. Hajiaghayi, R. Kleinberg, T. Leighton, and H. Räcke. Oblivious routing on node-capacitated and directed graphs. *Proc. of ACM-SIAM SODA*, 782–790, 2005.

[20] M.T. Hajiaghayi, G. Kortsarz, and M.R. Salavatipour. Approximating buy-at-bulk and shallow-light k-steiner tree. *Proc. of APPROX*, Springer LNCS 4110, 153–163, 2006.

[21] D. Hochbaum. *Approximation algorithms for NP-hard problems*. PWS Publishing Company, 1997.

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

[23] David S. Johnson. Approximation algorithms for combinatorial problems. *J. Comput. System Sci.*, 9:256–278, 1974.

[24] P. Klein and R. Ravi. A nearly best-possible approximation algorithm for node-weighted Steiner trees. *Journal of Algorithms*, 19(1):104–15, 1995.

[25] A. Meyerson, K. Munagala, and S. Plotkin. Cost-distance: two metric network design. *Proc. of IEEE FOCS*, 624–630, 2000.

[26] A. Moss and Y. Rabani. Approximation algorithms for constrained for constrained node weighted steiner tree problems. *Proc. of ACM STOC*, 373–382, 2001.

[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] A. Schrijver. *Combinatorial optimization*. Springer, 2003.

[29] K. Talwar. The single-sink buy-at-bulk lp has constant integrality gap. *Proc. of IPCO*, Springer LNCS, 475–486, 2002.

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