Approximation Algorithms for Non-Uniform Buy-at-Bulk Network Design [*]

C. Chekuri[†] M. T. Hajiaghayi[‡] G. Kortsarz[§] M. R. Salavatipour[¶]

Abstract

Buy-at-bulk network design problems arise in settings where the costs for purchasing or installing equipment exhibit economies of scale. The objective is to build a network of cheapest cost to support a given multi-commodity flow demand between node pairs. We present approximation algorithms for buy-at-bulk network design problems with costs on both edges and nodes of an undirected graph. Our main result is the first poly-logarithmic approximation ratio for the non-uniform problem that allows different cost functions on each edge and node; the ratio we achieve is $O(\log^4 h)$ where h is the number of demand pairs. In addition we present an $O(\log h)$ approximation for the single sink problem. Poly-logarithmic ratios for some related problems are also obtained. Our algorithm for the multi-commodity problem is obtained via a reduction to the single source problem using the notion of junction trees. We believe that this presents a simple but useful technique for other network design problems.

1 Introduction

Network design problems involve finding a minimum value (sub) network that satisfies various properties, often involving connectivity and routing between node pairs. Simple examples include spanning trees, Steiner trees, minimum cost maximum flow, 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. Often, the cost in a typical network design problem is some function of the chosen edges or nodes.

Buy-at-bulk network design problems arise in settings where economies of scale and the availability of capacity in discrete units result in concave or sub-additive[1] cost functions on the edges or nodes. One of the main application areas is in the design of telecommunication networks. The typical scenario is that capacity (or bandwidth) on a link can be purchased in some discrete units $u_1 < u_2 < \ldots < u_r$ with costs $c_1 < c_2 < \ldots < c_r$ such that the cost per bandwidth decreases $c_1/u_1 > c_2/u_2 > \ldots > c_r/u_r$. The capacity units are sometimes referred to as cables or pipes. The cables induce a monotone concave (or more generally a sub-additive) function $f : \mathbb{R}^+ \rightarrow \mathbb{R}^+$ where $f(b)$ is the minimum cost of cables of total capacity at least b. A basic problem that needs to be solved in this setting the following: given a set of bandwidth demands, install sufficient capacity on the links of an underlying network topology so as to be able to route the demands. Formally, we are given an undirected graph $G = (V, E)$ on n nodes that represents the network topology and a set of h demand pairs

[*]Most of the results of this paper appeared in preliminary form in two extended abstracts [8] and [9].

[†]Dept. of Computer Science, University of Illinois, Urbana-Champaign. Email: chekuri@cs.uiuc.edu. 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.

[1]A real valued function f is sub-additive if $f(x) + f(y) \geq f(x + y)$ for all $x, y \geq 0$.

$\mathcal{T} = \{s_1t_1, s_2t_2, \ldots, s_ht_h\}$. We refer to the end points of the pairs as terminals. Pair i has a non-negative demand $\delta(i)$. Routing of the demands consists of finding a feasible multi-commodity flow for the pairs in which $\delta(i)$ flow is sent from s_i to t_i. The objective is to minimize the value of the flow x_e is the total flow on edge e. In this paper we consider a more general problem where the function f can vary depending on the edge; that is for each edge $e \in E$ there is a given monotone sub-additive cost function $f_e : \mathbb{R}^+ \to \mathbb{R}^+$. The goal is still to find a minimum value feasible multi-commodity flow for the demands; the value of the flow in the more general setting is $\sum_{e \in E} f_e(x_e)$. We refer to this problem as MC (Multi-commodity Buy-at-Bulk).

We refer to the simpler case where the f_e is the same for all edges as the *uniform* problem and the general case as the *non-uniform* problem. An instance is called a single-sink (or single-source) instance if all the pairs have a common sink (source) s; the pairs are of the form $st_1, st_2, \ldots, st_k$ with s as the sink. We use SS to refer to such instances. A typical telecommunications problem with discrete capacity units gives rise to a uniform problem. However non-uniform cases arise often for several reasons including the following. First, not all capacity units are available at all links due to various constraints. Second, when designing networks incrementally, existing links can have different unused capacity available and this leads to non-uniformity.

Our discussion so far allowed costs on the edges of the network. A natural and useful generalization is to allow costs (or weights) on both *edges* and *nodes* of the graph. We are motivated to study this generalization by both theoretical as well as practical considerations. 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. Sometimes, costs on the nodes can be translated into costs on the edges in an approximate fashion to simplify the problems. However, this requires to work with directed graphs and 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. We can often easily reduce problems in which both the nodes and edges have costs to the case in which only nodes have costs. In this paper we consider the buy-at-bulk network design problem with costs on the nodes. We formally define it now. The input to the problem is the same as that for the edge-weighted case: an undirected graph G and a set $\mathcal{T} = \{s_1t_1, s_2t_2, \ldots, s_ht_h\}$ of h node pairs with each pair s_it_i specifying a non-negative demand $\delta(i)$. Each node $v \in V$ has a monotone sub-additive real valued function $f_v : \mathbb{R}^+ \to \mathbb{R}^+$ associated with it. Again, a feasible solution consists of finding a feasible multi-commodity flow for the pairs in which $\delta(i)$ flow is sent from s_i to t_i. We note that up to a ratio of $2 + \epsilon$, for any universal constant $\epsilon > 0$ we may assume that the that the flow in the solution is *unsplittable*. Namely, the solution may be a collection $P_1, P_2, \ldots, P_h$ of paths such that P_i connects s_i and t_i. See explanation later. Given the paths, $\delta(i)$ flow is routed along P_i for $1 \leq i \leq h$. The value 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 value. We refer to this problem as NMC (Node-weighted Multi-Commodity Buy-at-Bulk). The single-sink version is referred to as NSS.

We focus for the most part on NMC and NSS since they generalize MC and SS, respectively. Buy-at-bulk network design problems capture as special case some classical NP-hard connectivity problems such as minimum cost Steiner tree and Steiner forest problems. Unlike the above two problems that admit constant ratios [30, 4], it was shown by [1] that unless $NP \subseteq ZPTIME\left(n^{\text{Polylog } n}\right)$ the general buy-at-bulk network design problem admits no $\log^{1/2-\epsilon}$ approximation for any universal positive ϵ.

Our main result is the following.

Theorem 1.1 *There is a polynomial time $O(\log^4 h)$-ratio approximation algorithm for NMC, where h is the number of pairs.*

We present two algorithms for Theorem 1.1. One is based on rounding a solution to a linear programming relaxation. This algorithm has the advantage in giving an $O(\log^4 h)$ ratio even when the $D = \sum_i \delta(i)$ is super-polynomial in n and h. The other is a greedy combinatorial algorithm that yields a ratio of $O(\log^3 h \log D)$. The greedy algorithm is simpler (does not need to solve a complex LP) and purely combinatorial and is faster by many orders of magnitude than the LP-based algorithm.

Our algorithms for the multi-commodity problems are built upon their single-sink counterparts. For the edge-cost single-sink problem an $O(\log h)$-approximation was given by [28]. We give an algorithm for the node-cost version.

Theorem 1.2 *There is a deterministic $O(\log h)$-approximation algorithm for NSS where h is the number of terminals. Furthermore, the integrality gap of a natural linear programming relaxation for the problem is $O(\log h)$.*

An easy reduction from the set cover problem shows that unless P=NP, the node-weighted Steiner tree problem [24], a special case of NSS, cannot be approximated to a ratio better than $c \log h$ for some universal constant c. Thus the ratio guaranteed by Theorem 1.2 cannot be improved by more than a constant factor unless P=NP [29].

We also consider variations of NMC and NSS that require only a subset of pairs to be connected. Let $\mathcal{T} = \{s_1t_1, s_2t_2, \ldots, s_ht_h\}$. For a subset $\mathcal{T}' \subseteq \mathcal{T}$ we let $\text{OPT}(\mathcal{T}')$ denote the value of an optimum solution that connects the pairs in $\mathcal{T}'$. In the *density* problem, we seek to find a subset of pairs $\mathcal{T}'$ such that $\text{OPT}(\mathcal{T}')/|\mathcal{T}'|$ is minimized. A related problem is obtained when for a given integer parameter $k \leq h$ we seek to find $\mathcal{T}' \subseteq \mathcal{T}$ with $|\mathcal{T}'| = k$ such that $\text{OPT}(\mathcal{T}')$ is minimized. We use den-NMC and den-NSS (similarly den-MC and den-SS) to refer to the density versions of NMC and NSS (similarly MC and SS) and k-NMC and k-NSS (similarly k-MC and k-SS) for the k versions. Although these variants have been considered in the context of simpler network design problems before, they have been studied only recently in [20] for two-cost network design. In [20], the problem k-SS is referred to as the buy-at-bulk k-Steiner tree. We will prove the following theorem which in turn is used in the proof of Theorem 1.1.

Theorem 1.3 *There is a polynomial time $O(\log^2 h)$-approximation for den-NSS and an $O(\log^2 h \cdot \log D)$-approximation for k-NSS.*

1.1 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 [31] for classical results on polynomial time algorithms and to [18, 19, 34, 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.

Buy-at-bulk network design problems have been considered in both operations research and computer science in the context of flows with concave costs. The known results on approximation algorithms for buy-at-bulk network design are essentially for the edge-weighted problems. Salman et al. [32] were perhaps the first to consider approximation algorithms, in particular for the single-source version. For the uniform case of MC, Awerbuch and Azar [2] showed that the problem on an arbitrary graphs can be reduced to that on trees using probabilistic embedding of metric spaces in to tree metrics; using the best known distortion result [6, 13] yields an $O(\log n)$-approximation. Some special cases of the uniform MC admit constant factor approximation algorithms; Kumar et al. [25] and Gupta et al. [14] obtain constant factor approximation algorithms for the rent-or-buy problem where $f(x) = \min\{\mu x, M\}$. Constant-factor approximations are known also for the uniform single-source case via randomized combinatorial algorithms [16, 15] and an LP rounding approach [33]. For SS, an $O(\log h)$ randomized approximation was given first by Meyerson, Munagala and Plotkin [28]. In [11], the algorithm of [28] was derandomized using an LP relaxation - this also established an $O(\log h)$ integrality gap for the relaxation. For the multi-commodity problem the first non-trivial result is due to Charikar and Karagiazova

[10] who obtained an $O(\log D \exp(O(\sqrt{\log h \log\log h})))$-approximation. Hence our result is an exponential(!) improvement over the best known ratio, even for the simpler edge case. The question if the general demands (edge) case admits a polylogarithmic ratio was open ever since the [32] paper. Andrews [1] showed super-constant factor hardness of approximation for the multi-commodity versions: an $\Omega(\log^{1/4-\epsilon} n)$ factor for the uniform case and an $\Omega(\log^{1/2-\epsilon} n)$ for the non-uniform case. Chuzhoy et al. [7] showed an $\Omega(\log\log n)$ hardness for SS. These hardness results are based on the assumption that $\mathrm{NP} \not\subseteq \mathrm{ZPTIME}(n^{\mathrm{polylog}(n)})$. Table 1 summarizes the known results with the entries with a * indicating the results from this paper.

	Single Source		Multi-Commodity	
	Edge	Node	Edge	Node
Uniform	$O(1)$ [16] $\Omega(1)$	$O(\log h)$ * $\Omega(\log n)$	$O(\log n)$ [2] $\Omega(\log^{1/4-\epsilon} n)$ [1]	$O(\log^4 h)$ * $\Omega(\log n)$
Non-Uniform	$O(\log h)$ [28] $\Omega(\log\log n)$ [7]	$O(\log h)$ * $\Omega(\log n)$	$O(\log^4 h)$ * $\Omega(\log^{1/2-\epsilon} n)$ [1]	$O(\log^4 h)$ * $\Omega(\log n)$

Table 1: Approximation ratios and inapproximability for buy-at-bulk network design.

Organization: In the next section we present some notation used throughout the rest of the paper as well as an overview of the proofs. Section 3 presents the proof of Theorem 1.2. In Section 4 we present the approximation algorithm for NMC with arbitrary demands which uses LP rounding. This section also contains the proof of the existence of junction-trees as well as the proof of Theorem 1.3. Section 5 describes a greedy approximation algorithm for MC with polynomially bounded demands. Finally, we discuss some open problems and generalizations in Section 6.

2 Preliminaries

All graphs we consider are undirected. As mentioned earlier, we can easily transform problems with both node and edge costs into one in which only the nodes have costs 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 [3, 28]. This involves approximating the minimum value monotone sub-additive cost function f_v for each v by a collection of linear cost functions as follows. We assume without loss of generality that the demand values $\delta(i)$ for each pair $s_i t_i$ are non-negative integers. 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 $f_v^i : \mathbb{R}^+ \rightarrow \mathbb{R}^+$ 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) consists of a graph G and demand pairs $\mathcal{T} = \{s_1 t_1, s_2 t_2, \ldots, s_h t_h\}$. Each $s_i, t_i \in V$ has a demand $\delta(i) \geq 0$. We are given two separate functions $c : V \rightarrow \mathbb{R}^+$ and $\ell : V \rightarrow \mathbb{R}^+$; 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 value 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 value of the solution specified by V' is given as

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

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). [2]. The value of the flow is given by $\sum_{e \in E} f(x_e)$ where As just shown, the two-cost formulation shows that the optimum value for the unsplittable flow version of the problem is at most a constant factor more than the optimum value of a solution that allows the flow for each pair to be split among multiple paths. In the rest of the paper, we restrict our attention to the two-cost network design formulation of NMC and NSS (and similarly for MC).

Let $\mathcal{T}$ denote the set of source-sink pairs in the given instance and $h = |\mathcal{T}|$. The variable h' is used to denote the number of uncovered pairs remaining at some stage of the algorithm. If all demands $\delta(i)$ are equal then, by scaling down the demands and the costs, we can assume all demands are equal to 1. For this reason we refer to it as a unit-demand instance. We assume without loss of generality that each terminal is a node of degree 1 and that exactly one pair contains each terminal. This can be achieved by hanging dummy terminals.

In the rest of the paper, when we refer to an optimum solution to a given instance we assume some fixed optimum solution. We use OPT to denote its value. The optimum solution's fixed cost (i.e. first term in (1)), and length (second term in (1)) are denoted by OPT_c and OPT_ℓ, respectively. Note that by definition $\text{OPT} = \text{OPT}_c + \text{OPT}_\ell$. For a subset $V' \subseteq V$, the distance $\ell_{V'}(u, v)$ is the distance between u, v in the graph $G[V']$ induced by V'. If the graph $G[V']$ contains an s_i to t_i path, we say that V' *routes* or *covers* the pair s_i, t_i. The number of pairs routed in $G[V']$ is denoted by $\mathcal{T}(V')$. Assume $\mathcal{T}' = \mathcal{T}(V') \subset \mathcal{T}$ does not contain all the source-sink pairs and that $G[V']$ routes all the pairs of $\mathcal{T}'$ but no other pair. The (fixed) value and length (incremental value) of V' are $c(V') = \sum_{v \in V'} c_v$ and $R(V') = \sum_{i: s_i t_i \in \mathcal{T}'} \delta(i) \cdot \ell_{V'}(s_i, t_i)$, respectively. The value of partial solution V' is $\psi(V') = c(V') + R(V')$. The total *density* of partial solution V' is $\psi(V')/|\mathcal{T}'|$. We also define the cost density and length density of solution V' as $c(V')/|\mathcal{T}'|$ and $R(V')/|\mathcal{T}'|$, respectively.

We may drop some of the parameters in our notation if they can be deduced from the context. Unless specified differently all $\log$'s are in base 2. Our algorithms for NMC and MC are greedy iterative algorithms. In each iteration the algorithm finds a partial solution (a solution that routes some of the remaining uncovered demands) at low density, where the density is the ratio of the value of the partial solution to the number of new demands 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 $\mathcal{T}_i$ of the demands. Let* OPT *be the value 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 value 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 value of the solution returned by the algorithm is at most* $f(h) \cdot (\ln h + 1) \cdot \text{OPT}$.

2.1 Overview of Algorithmic Ideas

We briefly outline the high level ideas behind our algorithm for NMC. The algorithm follows a greedy scheme in an iterative fashion. In each iteration it finds a partial solution that connects a subset of the uncovered pairs The connected pairs are then removed. The *density* value of the partial solution is the ratio of the total value of the partial solution to the number of pairs in the solution. For some fixed constant a, the algorithm guarantees that the *density* of the partial solution it computes is at most $O(\log^a h) \cdot \text{OPT}'/h'$ where h' is the number of remaining

[2]To distinguish the cost of edges and the fixed cost from the total cost of the solution we use the word value to refer to the sum of fixed and the incremental costs

terminals and OPT$'$ is the value of an optimum solution for them. Using Lemma 2.1, this scheme yields an $O(\log^{a+1} h)$-approximation.

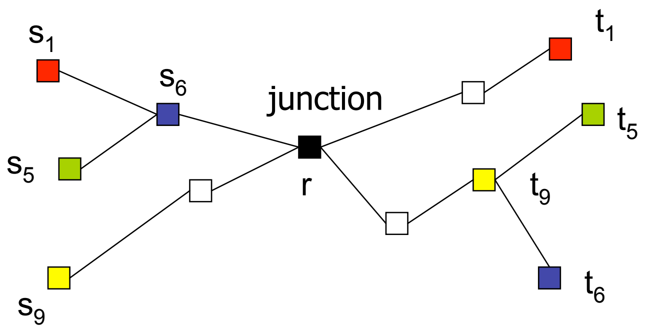

Figure 1: Junction tree for a subset of the pairs.

The key insight is to show the *existence* of a low-density partial solution that has a restricted structure. This structure allows us to find a near-optimal partial solution in polynomial time. The restricted structure of interest is what we call 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 for each pair in A, the unique path in T for the pair contains r. The value 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. We prove that given an instance of NMC 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, i.e. den-NSS. We use Theorem 1.2 and obtain an $O(\log^2 h)$-approximation for den-NSS 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. We observe that the above scheme effectively reduces the multi-commodity problem to the single-sink problem and this general paradigm is of broader applicability.

The first method uses an LP relaxation to solve the problem approximately. This LP is similar to the LP relaxation for SS proposed in [11]. Using the $O(\log h)$ upper bound on its integrality gap we obtain an $O(\log^2 h)$-approximation for den-SS and by a slight modification a similar ratio for finding the best density junction-tree.

We also present a combinatorial algorithm that is applicable when D is polynomial in h. In this section we also prove in a completely different way (than the proof for general demands) a junction tree lemma with better parameters (that unfortunately applies only to polynomial demands). Putting together these ingredients gives us the poly-logarithmic approximation for MC.

For NSS, our algorithm adapts the ideas in Klein and Ravi [24] for node-weighted Steiner tree with those of Meyerson et al. [28] for SS. The algorithm can be adapted to show an integrality gap for the natural linear programming relaxation by borrowing from Guha et al. [16] and Chekuri et al. [11].

3 The Node-Weighted Single-Sink Problem

In this section we prove Theorem 1.2. An instance of NSS in the two cost network design formulation consists of an undirected graph $G = (V, E)$ with a designated root node r, a set of terminals $T \subseteq V$, a demand function

$\delta : T \cup \{r\} \rightarrow \mathbb{R}^+$, a cost function $c : V \rightarrow \mathbb{R}^+$, and a length function $\ell : V \rightarrow \mathbb{R}^+$. The objective is to find an induced graph $G' = G[V']$ to minimize

$$c(V') + \sum_{t \in T} \ell_{G'}(r,t).$$

Recall that this problem admits approximation ratio of $O(\log h)$ for the case of edge costs edge-costs [28]. We give a similar ratio for node-costs (in general, problems with node costs seem to have been hardly studied earlier). The algorithm and especially the integrality gap proved for node costs can be later used to solve the density problem den-NSS. In turn, this solution is used in the solution for NMC.

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 $+\infty$); this will subsequently help simplify the description of our algorithm. If for a node v, $c(v) = \ell(v) = 0$ then we can add this node to any solution at no costs. So, without loss of generality, we may further assume that for every node v, either $c(v) > 0$ or $\ell(v) > 0$.

A *spider* is a connected tree with at most one node of degree more than two; In other words a spider is a collection of paths that are pairwise vertex disjoint, except that they share the start vertex. This vertex is called the *center* of the spider. If the spider has a node of degree at least three, its center is unique. Every leaf of a spider must be a terminal. The density of a spider is the ratio of its total value over the number of terminals in the union of its leaves and its center, where the total cost depends on the problem definition. Spiders were defined and used in an iterative greedy algorithm for the node-weighted Steiner tree problem by Klein and Ravi [24]. We use spiders for NSS by generalizing the cost of a spider appropriately and also by using randomization in a crucial way following the algorithm of Meyerson et al. [28] for SS. We then use the ideas in [17] and [11] for node-weighted Steiner tree and SS respectively to also obtain an integrality gap for a natural LP relaxation.

A randomized algorithm for NSS: For the ease of exposition, first we describe a randomized algorithm for NSS that is inspired by the spider approach of [24] and the randomized merging algorithm of [28] for (the edge-weighted) SS. To describe the algorithm we first define the value of a spider in the setting of NSS. 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 set of terminals at the leaves of the spider. For a terminal $t \in T(\mathcal{S})$ we let p_t denote the path between t and the center of the spider $\mathcal{S}$. Although the definition of a spider requires the paths p_t, for $t \in \mathcal{S}$, to be internally node-disjoint, we abuse notation and allow the paths to share nodes. This will not affect the cost-density because we are comparing against the density of a spider, namely if the best density spider has center r and k terminal leaves, we may form a tree rooted from the center r and the k shortest paths from r to the terminals. The resulting structure is a tree (not a spider) of value no larger than the value of the spider. Thus we can think of a spider $\mathcal{S}$ as prescribed by a center s, a set of terminals $T(\mathcal{S})$ and a path p_t from each $t \in T(\mathcal{S})$ to s. The total value of a spider $\mathcal{S}$ with center s, denoted by $\beta(S)$, is:

$$c(s) + \sum_{t \in T(\mathcal{S})} (c(p_t) - c(s) + \delta(t) \cdot \ell(p_t)), \tag{2}$$

where $c(p_t)$ and $\ell(p_t)$ are the sum of the costs and lengths of the nodes on p_t, respectively. Note that if the p_t's are not internally node disjoint then the value of the spider would count the cost of a shared node multiple times. The randomized algorithm RandSpider is described in Fig 2.

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 spider $\mathcal{S}$ - 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 [28] that RandSpider yields a solution of expected value $O(\log h \cdot \text{OPT})$ 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.

Algorithm RandSpider for NSS:

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 2: A randomized algorithm for NSS

For a terminal t and a node v we let $d_t(v)$ denote $\min_{p\in P_{tv}}(c(p) + \delta(t)\cdot \ell(p))$ 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 $c(u) + \delta(t) \cdot \ell(u)$.

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

Proof. Let s be the center of a minimum density spider. For each node $v \in V$ we run an algorithm to be described below that computes a minimum density spider with center v and thus we can assume that we know s. For simplicity, we assume that s is not a terminal - this can always be ensured by hanging dummy terminal. 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 corresponds to distance $d_{t_i}(s)$. 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 $j^* = \text{argmin}_j \alpha_j$. We return the subgraph $\mathcal{S}$ obtained by the union of the paths $P_1, P_2, \ldots, P_{j^*}$. We show the density of $\mathcal{S}$ is no more than the density of a minimum density "real" spider (by real spider we mean a tree with internally disjoint nodes). Say that the best density spider has j leaves. Note that j is one of the "guesses" of number of terminals used by the algorithm. The density of our tree is no larger than the density of the spider as we the j shortest paths from s to the terminals □

A linear programming relaxation for NSS: We first formulate NSS as an IP for which we have the following LP relaxation. For $t \in T$, let $\mathcal{P}_t$ denote the set of paths from root r to t. We assume that the terminals are at distinct nodes (we can easily enforce this by replacing multiple terminals by a single new terminal whose demand is the sum of the original terminals' demand) and hence $\mathcal{P}_t$ and $\mathcal{P}_{t'}$ are disjoint. 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:

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

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

Let OPT_{LP} be the value of an optimum solution to LP-NSS. We prove that RandSpider yields an integral solution of expected value $O(\log h \cdot \text{OPT}_{LP})$. The proof of the following lemma uses ideas from [17] and is deferred to Appendix A.

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

We assume the lemma and prove Theorem 1.2. We will show how to derandomize RandSpider via the linear programming relaxation using ideas similar to [11]. 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 the linear program LP-NSS on instances 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 value 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 value 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 $\delta(t) \cdot \alpha(t)$ 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 $\delta(T(S)) \cdot \alpha(t)$. Thus it can be seen that the expected value 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 value as in Equality 2

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

Proof. We can bound the expected value as follows. The value 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 at most $\beta(\mathcal{S})$, 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 $\sum_{t \in T(\mathcal{S})} a_t \cdot \delta(T(\mathcal{S})) \cdot \ell(p_t)$ 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 value is at most $2\beta(\mathcal{S})$. □

Proof of Theorem 1.2. We first prove via induction on h that RandSpider yields a solution of *expected* value at most $3H_h \cdot \text{OPT}_{LP}$ 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. If $h = 1$ then it can be easily checked that the algorithm returns an optimum solution. Furthermore, the case $h = 1$ is the problem of finding minimum distance between two pairs (by the values defined above that include both the fix and incremental costs) and thus, the integrality gap is 1. [3]

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 value of the solution produced by RandSpider to I' is at most $3H_{h'} \cdot \text{OPT}_{LP}(I')$ where $h' = h - k + 1$ is the number of terminals in I'. Using Lemma 3.3, this

[3]Am I right, and should we cite?

expected value is at most $3H_{h'} \cdot \text{OPT}_{LP}(I)$. The total value of the solution for I is the sum of two costs: (i) the cost of the solution to I' and (ii) the cost of the routing of non-proxy terminals in $\mathcal{S}$ to the chosen proxy terminal. The expected cost of (ii) is, by Lemma 3.4, bounded by $2\beta(\mathcal{S})$. Using Lemma 3.2 and the fact that $\mathcal{S}$ is a minimum density spider with k terminals, we have that $2\beta(\mathcal{S}) \leq 2k\text{OPT}_{LP}(I)/h$. Putting together these observations and using linearity of expectation, the expected value of the solution to I is at most

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

The algorithm RandSpider can be derandomized using a solution to LP-NSS as described below. The argument is essentially the same as the one in [11]. 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 value 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. Note that α_t and β_t can be computed in polynomial time from x^*, f^* and $\mathcal{S}$. Let $t' = \arg\min_t(3H_{h'}\alpha_t + 2\beta_t)$. The above probabilistic analysis shows that $3H_{h'} \cdot \alpha_{t'} + 2\beta_{t'} \leq 3H_h \cdot \text{OPT}_{LP}(I)$. 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 value of the solution on $I'_{t'}$ is at most $3H_{h'} \cdot \alpha_{t'}$. Therefore the total value of the solution is $3H_{h'} \cdot \alpha_{t'} + 2\beta_{t'} \leq 3H_h \cdot \text{OPT}_{LP}(I)$ as desired. □

4 The Node-Weighted Multi-Commodity Problem

In this section we prove Theorems 1.1 and 1.3. The general structure of the algorithm of Theorem 1.1 is as the outline described in Subsection 2.1. It follows an iterative greedy scheme. In each iteration we find a partial solution that connects a subset of the pairs that are yet uncovered. The connected pairs are then removed. The *density* of the partial solution is the ratio of the total value 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 poly-logarithmic (specifically $O(\log^3 h)$) factor away from the density of the optimum solution. By Lemma 2.1 an $O(\log^4 h)$ ratio follows for the MC instance. The rest of this section is devoted to showing how to find a partial solution with density $O(\log^3 h)$.

We first prove that given an instance of NMC there is always a is a junction-tree of density $O(\log h)$ times the optimum density.

It may seem that as we know how to approximate within $O(\log h)$ the *rooted* version of MC, we can use this to directly find a good junction tree. However, there is a major difficulty. We dont know the root and the participating pairs in the junction tree.

Hence we use LP techniques (see the proof of Theorem 1.2). We approximate the density of the best junction tree by $O(\log^2 h)$. The problem of finding a low density junction-tree is closely related to the density variation of NSS, called den-NSS in which we want to find a solution with minimum density i.e. the ratio of total value over the number of terminals spanned (v.s. the total value as in SS). We use Theorem 1.2 and obtain an $O(\log^2 h)$-approximation for den-NSS and by a slight modification a similar ratio for finding a minimum density junction-tree.

Note that in total we loose a $O(\log^3 h)$ factor; An $O(\log^2 n)$ loss is because this is the approximation ratio for the density of a junction tree, and a further $O(\log h)$ loss because the best density junction tree may have density $O(\log h)$ worse than the optimum. This proves Theorem 1.3 and also allows us to find a junction tree of density $O(\log^3 h) \cdot \frac{\text{OPT}}{h}$. Then we remove the pairs that are connected and iterate in a greedy fashion. This results in an approximation ratio of $O(\log^4 h)$ for NMC by a set-cover like analysis.

4.1 Junction Tree Lemma for general demands

We prove the following lemma on the existence of a junction tree with low density.

Lemma 4.1 *Given an instance of MC 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. In [8] proofs are given for two lemmas with slightly weaker bounds and a proof idea that combined aspects of both those lemmas was suggested by Harald Räcke. We need the following technical lemma first.

Lemma 4.2 *Given an instance of NMC 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 $\min(n, h^2)$.*

Proof. We have a trivial upper bound of n on the number of degree 2 nodes thus we focus on proving the bound of h^2. We assume without loss of generality that the terminals are all distinct; we can use dummy terminals as necessary. Consider an optimum solution $G[V^*]$. Each pair s_it_i uses a shortest ℓ-node-weighted path P_i in $G[V^*]$ to route its demand. We can assume that P_i is the unique shortest path between s_i and t_i; this can be arranged by considering a lexicographical ordering of the nodes and edges of G. Therefore, for any two distinct pairs s_it_i and s_jt_j, $P_i \cap P_j$ is 1-connected. [4] Thus the two paths may meet at some node, share a subpath for a while, and then be separated and never meet again. Now consider adding the paths $P_1, P_2, \ldots, P_h$ in order. Consider a path P_i. By the above observation, P_i can add 2 vertices of degree 3 in any previous path P_j, for every $j < i$. P_i creates one vertex of degree 3 when joining P_j, and another vertex of degree 3 when departing from P_j for (and no more since the path do not intersect again). Thus P_i can create at most $2(i-1)$ nodes of degree 3. Thus the total number of nodes with degree strictly greater than 2 is bounded by $\sum_{i=1}^{h} 2(i-1) \leq h^2$. $\square$

Before we can prove the junction tree theorem we need to prove the following lemma. 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 G and a set of node pairs 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 $|A|/4$.*

The next discussion is dedicated to proving this lemma.

Given an instance of NMC 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 embedding of graphs into trees [5], 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

[4]We thank Anupam Gupta for making this observation in answering a question about spanners.

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.

Lemma 4.4 *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 [5], Bartal created a distribution of laminar families that yields a probabilistic embedding of a graph metric into dominating 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 [5]. 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 $\min\{1, c \log n \cdot d_G(a, b)/\Delta\}$. The laminar family for G is obtained by applying the partitioning procedure recursively to the graphs $G = G_1, G_2, \ldots, G_k$. 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 $c \geq 4c'$. □

Lemma 4.4 implies Lemma 4.3. Now we are ready to 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 Lemma 4.3 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. Using T we create junction trees $T_1, T_2, \ldots, T_k$ with roots $r_1, r_2, \ldots, r_k$ that satisfy the following properties.

- Each node $v \in V^*$ is in $O(\log h)$ junction trees.
- For each $s_i t_i \in \mathcal{T}'$ there is some $1 \leq j \leq k$ such that $\ell_{T_j}(r_j, s_i) + \ell_{T_j}(r_j, t_i) \leq O(\log h)\ell_{G^*}(s_i, t_i)$.

Assuming the properties, we claim that one of the junction trees has density $O(\log h)\text{OPT}/h$. To prove the claim we assign each pair in $\mathcal{T}'$ to a unique tree that satisfies the second property above. Now we compute the total value of all the junction trees which consists of the total fixed cost and total incremental costs. From the first property the total fixed cost of all junction trees is at most $O(\log h)c(V^*) \leq O(\log h) \cdot \text{OPT}_c$. From the second property and the assignment of each pair to a unique tree, the total incremental cost of all junction trees is $O(\log h) \sum_{s_i t_i \in \mathcal{T}'} \ell_{G^*}(s_i, t_i) = O(\log h)\text{OPT}_\ell$. Thus the total value of all junction trees is $O(\log h)(\text{OPT}_c + \text{OPT}_\ell) = O(\log h)\text{OPT}$. Since $|\mathcal{T}'| \geq |\mathcal{T}|/4$, there is a junction tree in $T_1, T_2, \ldots, T_k$ of density at most $O(\log h)\text{OPT}/h$. This finishes the proof of the claim.

To obtain the junction trees 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. We observe that the paths are node disjoint. Let u_j and r_j be the internal node and the leaf end points of P_j respectively. Let $H_j = G^*_{u_j}$. We call each H_j a cluster and we call r_j its center.We create a junction tree T_j in each cluster H_j as follows. We let T_j be the shortest path tree in H_j rooted at r_j. We assign a pair $s_i t_i \in \mathcal{T}'$ to T_j if and only if the least common ancestor of s_i and t_i in T belongs to P_j. We now prove that the junction trees satisfy the two desired properties.

Consider an arbitrary node $a \in V^*$. For the first property, suppose a is in the trees $T_{j_1}, T_{j_2}, \ldots, T_{j_m}$ where $1 = j_1 < j_2 < \ldots j_m$. We observe that the number of leaves in T_{j_c} is no more than half the leaves in $T_{j_{c-1}}$ because the path constructed from $T_{j_{c-1}}$ starts at its root and picks the child with the heaviest number of leaves at each step. Thus the number of trees containing a is $O(\log h)$ since the total number of leaves in T is $\min\{n, h^2\}$.

For the second property, suppose we assign $s_i t_i \in \mathcal{T}'$ to T_j. Since $s_i t_i$ is $O(\log h)$-good, it follows that the diameter of $H_j = G^*_{u_j}$ is $O(\log h) d_{G^*}(s_i, t_i)$. Since $r_j \in V(G^*_{u_j})$, $d_{H_j}(r_j, s_i) = O(\log h) \cdot d_{G^*}(s_i, t_j)$ and $d_{H_j}(r_j, t_i) = O(\log h) \cdot d_{G^*}(s_i, t_i)$. Since $\ell_H(a, b) \leq d_H(a, b) \leq 2\ell_H(a, b)$ for any two nodes a, b and any subgraph H, we obtain the desired property. This proves the lemma. □

4.2 Finding an approximate min-density junction tree

In this subsection we prove Theorem 1.3. Specifically, we give an $O(\log^2 h)$-approximation algorithm for den-NSS and min-density junction tree. We also show how to obtain an $O(\log^2 h \cdot \log D)$-approximation for k-NSS. The algorithms and analysis are built upon the LP relaxation and the proof of the integrality gap for NSS 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 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 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 not. We normalize $\sum_t y_t$ to 1.

LP-NSSD:

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

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

Proposition 4.5 *For a given instance of den-NSS, let α^* be the density of the minimum density tree and let α be the optimum value of LP-NSSD. Then $\alpha \leq \alpha^*$.*

Proof. Let H be an optimum solution to the given instance of den-NSS and let $T' \subseteq T$ be the terminals connected to r. For $t \in T'$ let p_t be the path in H from t to r. The total value of routing is $c(V(H)) + \sum_{t \in T'} \delta(t) \ell_H(r, t)$. Therefore $\alpha^* = \frac{1}{k}(c(V(H)) + \sum_{t \in T'} \delta(t) \ell_H(r, t))$ where $k = |T'|$. We show a feasible solution to LP-NSSD as follows. For each $t \in T'$ we set $y_t = 1/k$. For each $v \in V(H)$ we set $x_v = 1/k$. For each t we set $f(p_t) = 1/k$. The other variables are set to 0. It is easy to check that this yields a feasible solution to LP-NSSD of value α^* and hence $\alpha \leq \alpha^*$. □

Theorem 4.6 *There is an $O(\log^2 h)$-approximation for den-NSS.*

Proof. Given an instance of den-NSS, obtain an optimum solution to LP-NSSD and let its value be α. For $p = 1 + 2\lceil \log h \rceil$ we obtain disjoint subsets of the terminals $T_1, T_2, \ldots, T_p$ as follows. Let $y_{\max} = \max_t y_t$. For $0 \le a \le 2\lceil \log h \rceil$, let $T_a = \{t \mid y_{\max}/2^{a+1} < y_t \le y_{\max}/2^a\}$. Since $\sum_{t \in T} y_t = 1$ there is an index b such that $\sum_{t \in T_b} y_t \ge 1/p$. From this we also have that $|T_b| y_{\max}/2^b \ge 1/p$. We now solve an NSS instance on T_b using the algorithm from Theorem 1.2. We claim that the resulting solution is an $O(\log^2 h)$-approximation to den-NSS. To prove this, we observe that scaling up, by a factor of $2^{b+1}/y_{\max}$, the given optimum solution to LP-NSSD yields a feasible solution to LP-NSS on the terminal set T_b. The value of this scaled solution to LP-NSS is $2^{b+1} \cdot \alpha / y_{max}$. Since the integrality gap of LP-NSS is $O(\log h)$ (by Theorem 1.2), we obtain an integral solution that connects each terminal in T_b to the root such that value of the solution is $O(\log h) \cdot 2^{b+1} \cdot \alpha / y_{\max}$. The density of this solution is therefore $O(\log h) \cdot 2^{b+1} \cdot \alpha / (y_{\max} |T_b|)$ which is $O(\log h) \cdot 2p\alpha$. Since $p = O(\log h)$ the density is $O(\log^2 h) \cdot \alpha$. Using Proposition 4.5, we obtain an $O(\log^2 h)$ approximation for den-NSS and also the same bound on the integrality gap of LP-NSSD. $\square$

Corollary 4.7 *There is an $O(\log^2 h)$-approximation for computing the minimum density junction tree.*

Proof. Recall that we can transform a given instance of NMC into one in which each terminal participated in exactly one pair. We obtain an instance of rooted den-NSS by letting $T = \{s_1, t_1, s_2, t_2, \ldots, s_h, t_h\}$ and guessing the root r of a minimum density junction tree. If we simply use the $O(\log^2 h)$-approximation guaranteed by Theorem 4.6 on this instance of den-NSS, we may not even get a feasible junction tree; the solution may include only one of the end points for each pair. To overcome this we solve LP-NSSD on the den-NSS instance on r and T with some additional constraints. For each pair $s_i t_i$ we add the constraint: $y_{s_i} = y_{t_i}$. The proof of Proposition 4.5 can be easily extended to show that the linear program with these additional constraints is a valid relaxation for the minimum density junction tree problem. We then apply the same rounding procedure as the one in the proof of Theorem 4.6. It can be seen that the new constraints ensure that for each pair $s_i t_i$ either we connect both s_i and t_i to r or neither of them. We can use essentially the same proof as that of Theorem 4.6 to the new setting to show that the algorithm yields an $O(\log^2 h)$-approximation for the minimum density junction tree problem. $\square$

Corollary 4.8 *There is an $O(\log^2 h \log D)$-approximation for k-NSS.*

Proof Sketch. We prove a bound of $O(\log^2 h \log k)$ for the simpler case when $\delta(i) = 1$ for each terminal. The algorithm for den-NSS can be applied to obtain an algorithm for k-NSS as follows. We compute a minimum density tree T using the $O(\log^2 h)$-approximation. If T contains $k' < k$ terminals, we remove the terminals contained in T from the set of terminals and recursively solve a residual problem to connect $k - k'$ terminals from the remaining terminals. If $k' > k$ we need to prune T to find a subtree of k terminals of density comparable to that of the tree that can be connected to the root. Using simple averaging arguments it is easy to find a subtree T' of T such that T' contains k terminals and such that density of T' is no more than a constant factor times that of T. However we need to connect the root r' of T' to the root r of T to obtain a feasible solution to k-NSS. We may not be able to pay for the path connecting r' to r while maintaining the density bound. To be able to pay this without increasing the value of the solution by too much, we preprocess the nodes in the graph to eliminate those that are "too far" from r. We do this as follows. We first guess a number OPT' which is within a constant factor of OPT. Let V' be the set of nodes $u \in V(G)$ such that u has a path p to r with the property that $c(p) + k\ell(p) \le \text{OPT}'$. Then we run the density algorithm in $G[V']$. Note that there is an optimum solution in $G[V']$ of value OPT. Thus we run the density algorithm in $G[V']$ which ensures that T' can be connected to r while increasing the value by only $O(\text{OPT})$.

The total value of this algorithm for k-NSS can be bounded by $O(\log^2 h \log k)\text{OPT}$ where the additional $\log k$ factor comes from the recursive step to greedily augment to k terminals when $k' < k$. This is similar to the standard set cover type analysis. For the arbitrary demand case, we get a ratio of $O(\log^2 h \log D)$ by considering $\delta(i)$ terminals in place of a terminal of demand $\delta(i)$. We omit further details. $\square$

4.3 Proof of Theorem 1.1

We put together the necessary ingredients to prove Theorem 1.1. As described in Section 2.1 the algorithm for NMC works in iterations. At the beginning of iteration i there is a residual problem to route the pairs $\mathcal{T}_i \subseteq \mathcal{T}$ with $\mathcal{T}_1 = \mathcal{T}$. In iteration i the algorithm finds an approximation for the minimum density junction tree for the pairs $\mathcal{T}_i$ using the algorithm from Corollary 4.7. Let $\mathcal{T}_i'$ be the pairs routed by the tree returned by the junction tree algorithm. We set $\mathcal{T}_{i+1} = \mathcal{T}_i \setminus \mathcal{T}_i'$ and the algorithm stops when $\mathcal{T}_{i+1} = \emptyset$. Since the junction tree routes at least one pair, $|\mathcal{T}_i'| > 0$ in each iteration and hence the algorithm terminates in at most h iterations. The total value of the solution can be bounded as follows. In iteration i there is a solution of value OPT to route $\mathcal{T}_i$ since $\mathcal{T}_i \subseteq \mathcal{T}$. From Lemma 4.1 and Corollary 4.7, the density value of the junction tree that routes the pairs in $\mathcal{T}_i'$ is $O(\log^3 h) \cdot |\mathcal{T}_i'| \cdot \text{OPT}/|\mathcal{T}_i|$. Applying Lemma 2.1, the total value of all the junction trees is $O(\log^4 h)\text{OPT}$.

In each iteration the algorithm finds an approximate junction tree. The running time for this is dominated by the time required to solve the linear program LP-NSSD. We need to solve this linear program n times since we have to guess the root. Each solution to the linear program is followed by a rounding phase which requires running the RandSpider algorithm. Overall the running time in each iteration is $O(nA + n^2hB)$ where A is the time to solve LP-NSSD and B is the time to compute a minimum density spider rooted at a given center. By computing all pairs shortest paths in advance, the running time to compute a minimum density spider can be reduced to $O(h)$ time. Thus each iteration can be implemented in $O(nA + n^2h^2)$ time with $O(n^3 \log D)$ preprocessing. The additional factor of $\log D$ is required since we need to compute shortest paths for different demand values at each terminal. The number of iterations is $O(h)$ and hence we obtain a running time of $O(nhA + n^2h^3)$.

5 A Greedy Approximation Algorithm for Polynomial Demand MC

In this section we mainly focus on the the edge-weighted version of multi-commodity buy-at-bulk, i.e. MC and describe a *greedy* combinatorial algorithm for MC that has an approximation ratio of $O(\log^3 h \log D)$, where D is the total demands of all the pairs. The algorithm can be adapted to work for NMC as well to achieve the same asymptotic ratio. This is discussed at the end of this section.

The overall structure of the algorithm is similar to the one presented in Section 4, i.e. it runs in iterations and in each iteration it greedily finds a partial solution with good density. The partial solution is a junction tree. Here, we give another junction tree lemma (with a different proof) which has better guarantees. The proof of this lemma is used in the analysis of the greedy algorithm we present later.

We now work in the edge-weighted setting and in the two-cost formulation each edge has a cost c_e and a length ℓ_e. The objective is to find $E' \subseteq E$ with $G' = G[E']$ to minimize the value

$$c(E') + \sum_{i=1}^{h} \delta(i) \cdot \ell_{G'}(s_i, t_i), \tag{3}$$

where $c(E') = \sum_{e \in E'} c_e$ and $\ell_{G'}(u, v)$ is the shortest ℓ-edge-weighted path distance between u and v in G'. As before, for an optimum solution for the given instance, the total cost, (fixed) cost, and length (incremental cost) are denoted by OPT, OPT_c, and OPT_ℓ, respectively; and $\text{OPT} = \text{OPT}_c + \text{OPT}_\ell$.

5.1 An improved junction tree lemma for D polynomial in h

The following junction tree lemma is better than Lemma 4.1. Unfortunately it works only for demands polynomial in n because our first step is to reduce the demands to unit demands. Below, wherever we use the terms distance or length or diameter it is with respect to the length (incremental cost) function ℓ.

Lemma 5.1 *Given an instance of MC with unit demands there is a junction-tree of cost density $O(\text{OPT}_c/h)$ and diameter $O(\log h)\cdot\frac{\text{OPT}_\ell}{h}$. For the general case with total demand D, there exists a junction-tree with cost density $O(\log h)\cdot\frac{\text{OPT}_c}{D}$ and diameter $O(\log h)\cdot\frac{\text{OPT}_\ell}{h}$.*

We first restrict our attention to the case of unit demands. By reducing the general demand case to the unit demand case by duplicating terminals, it follows that there is a junction-tree of density $O(\log D)\frac{\text{OPT}}{D}$. We later show that we can prove a stronger bound of $O(\log h)\frac{\text{OPT}}{D}$.

In the rest of this subsection we prove Lemma 5.1. Consider an optimum solution E^* to the given instance and let $G^* = G[E^*]$. Define $L = \sum_i \ell_{E^*}(s_i,t_i)/h = \text{OPT}_\ell/h$ to be the average length of the pairs in the optimum solution. In the following we assume the knowledge of E^* and hence we only prove the existence of the junction tree. We give an algorithm to decompose G^* into connected node-disjoint induced subgraphs $G_1 = G[V_1],\ldots,G_k = G[V_k]$ and also associate with each G_i a subset of pairs $\mathcal{T}_i'$ with both end points in G_i. This decomposition has several properties that we describe next. Let $\mathcal{T}' = \bigcup_i \mathcal{T}_i'$ be the set of pairs that are *preserved* in the decomposition. Any other pair is *lost*.

Lemma 5.2 *There is a decomposition of G^* into connected node-disjoint induced subgraphs $G_1 = G[V_1],\ldots$, $G_k = G[V_k]$ and associated disjoint subsets of the pairs $\mathcal{T}_1',\ldots,\mathcal{T}_k'$ such that:*

1. *The total number of preserved pairs $|\mathcal{T}'| \geq h/8$.*

2. *For $1 \leq i \leq k$, the diameter of G_i is at most $\Delta = 2\log h\cdot L$.*

3. *For each pair s_jt_j in $\mathcal{T}_i'$, $\ell_{G_i}(s_j,t_j) \leq 2L$.*

4. *For $1 \leq i \leq h$, G_i has low cost density, that is, $c(G_i)/|\mathcal{T}_i'| \leq 8\text{OPT}_c/h$.*

We prove Lemma 5.2 using several claims.

First we prune the pairs whose shortest paths are large compared to L. The claim below follows from a simple averaging argument.

Claim 5.3 *The number of pairs s_jt_j such that $\ell_{E^*}(s_j,t_j) \geq 2L$ is at most $h/2$.*

We restrict attention to those $h/2$ pairs s_jt_j such that $\ell_{E^*}(s_j,t_j) \leq 2L$. For each pair s_jt_j we fix a shortest ℓ-path Q_j in G^*. For a subgraph H of G and a node $u \in V(H)$ we let $B_H(u,r)$ be the set of all nodes in H at ℓ-distance at most r from u; we call this the *sphere* with center u and radius r. We abuse notation and use $B_H(u,r)$ also to denote the graph induced by the nodes and the edges of the sphere. A pair s_jt_j is said to *touch* a sphere if some node of path Q_j belongs to the sphere. A pair s_jt_j that touches the sphere is *inside* the sphere if all the nodes of Q_j are in the sphere. Let $g_H(u,r)$ be the number of pairs that are inside $B_H(u,r)$ and let $g'_H(u,r)$ be the number of pairs that touch $B_H(u,r)$. We drop H when the graph in question is clear. We obtain the decomposition from G^* as follows. For $i \geq 1$ let $r_i = i\cdot 4L$. Pick an arbitrary source v and consider the graphs $B(v,r_i)$ for $i \geq 1$. Let j be the least index such that $g(u,r_j) \geq g'(u,r_j)$ (note that a pair which touches sphere $B(v,r_i)$ will be inside of sphere $B(v,r_{i+1})$). We set $G_1 = B(u,j\cdot 4L)$. We now recurse on the graph $G^* - G_1$ after we remove all the pairs that touch G_1. The recursion stops when there are no pairs left in the graph. Note that a pair that touches G_1 but is not inside G_1 is not retained in the decomposition. Such a pair is said to be lost.

Claim 5.4 *The radius of G_1 is at most $(\log h\cdot L)$; so the diameter is at most $\Delta = 2\log h\cdot L$.*

Proof. Recall that $G_1 = B(u, r_j)$; therefore it is sufficient to prove that $j \leq \log h$. From the choice of j it follows that for each $i < j$: $g(u, r_i) < g'(u, r_i)$. We note that a pair that touches $B(u, r_i)$ is inside $B(u, r_{i+1})$ because we assumed the distance between every pair is at most $2L$; thus for $i < j$: $g(u, r_{i+1}) \geq 2g(u, r_i)$. The total number of pairs is $h/2$ and hence $j \leq \log h$. $\square$

Claim 5.5 *The number of lost pairs in the overall decomposition is at most $h/4$.*

Proof. When G_1 is created the pairs that are lost are those that touch G_1 but are not inside. By construction the number of these pairs is at most the number of pairs inside G_1. Thus we can charge the lost pairs to those retained in G_1. By Claim 5.3 there were a total of at least $h/2$ pairs. $\square$

Now discard every subgraph (sphere) G_i for which the cost density is larger than $8\text{OPT}_c/h$ and let $\mathcal{S} = \{G_1, \ldots, G_k\}$ be the set of remaining subgraphs; $\mathcal{S}'$ is the set of discarded subgraphs. Observe that:

$$\sum_{G_j \in \mathcal{S}'} \frac{8\text{OPT}_c \cdot \mathcal{T}'_j}{h} \leq \sum_{G_j \in \mathcal{S}'} c(G_j) \leq \text{OPT}_c.$$

The last inequality follows as the subgraphs are node-disjoint and therefore edge-disjoint. This implies that the number of pairs in the subgraphs discarded (i.e. in $\mathcal{S}'$) is at most $h/8$. Therefore:

Claim 5.6 *The number of pairs in the subgraphs in $\mathcal{S}$ is at least $h/8$.*

Claims 5.3 to 5.6 show the existence of the desired decomposition for Lemma 5.2.

Using Lemma 5.2, we show that there is a junction-tree of cost density $O(\frac{\text{OPT}_c}{h})$ and length density $O(\log h)\frac{\text{OPT}_\ell}{h}$. In each G_i pick an arbitrary node v_i and let T_i be a shortest path tree in G_i rooted at v_i. Let E_i be the edge-set of T_i. Note that $E' = \cup_i E_i$ is a partial solution for the pairs in $\mathcal{T}'$ and $E' \subseteq E^*$. By the diameter guarantee, the distance from any node in G_i to v_i is at most Δ. Note that T_i is a candidate junction-tree for the pairs in G_i. We claim that one of these junction trees has the desired density. To prove this we compute the total cost of these k junction-trees. The sum of (fixed) cost is $\sum_{i=1}^{k} c(E_i) \leq c(E^*)$ and the number of pairs in $\mathcal{T}'$ is at least $h/8$ (by Lemma 5.2) and hence one of the trees has cost density no more than $O(\frac{\text{OPT}_c}{h})$; also by the diameter guarantee in Lemma 5.2, is at most Δ and so the length density is no more than $O(\log h) \cdot \frac{\text{OPT}_\ell}{h}$.

We now consider the case of arbitrary D. Again, by averaging there exists a junction-tree of cost density $O(\frac{\text{OPT}_c}{D})$. However, we claim a diameter bound of $O(\log h \cdot L)$ in each of the G_i instead of $O(\log D \cdot L)$. To obtain this bound we modify the choice of v in creating each sphere G_i (see proof of Lemma 5.2). Instead of picking an arbitrary source point, we pick a source v to be the one with the largest demand (that is largest demand before duplications) among the remaining pairs. This ensures that the index j in the proof of Claim 5.4 remains $O(\log h)$ since $\max_j d_j/D \geq 1/h$. This finishes the proof of Lemma 5.1.

5.2 The Greedy Approximation Algorithm

The algorithm relies on a result of [20] regarding shallow-light trees (described below). The instance to the shallow-light k-Steiner problem is a graph $G(V, E)$, with edge-weight function $c : E \rightarrow \mathbb{R}^+$ and edge-length function $\ell : E \rightarrow \mathbb{R}^+$, a collection T of terminals containing a root s, a positive integer k, and a diameter bound L. The goal is to find an s-rooted k-Steiner tree that has ℓ-diameter at most L, and among all such subtrees, find the one with minimum c-cost. A (ρ_1, ρ_2)-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 following theorem is from [20].

Theorem 5.7 *[20] There exist two universal constants c_1, c_2 and a polynomial time algorithm $\mathcal{A}$ for which the following holds. Consider an instance of shallow-light k-Steiner as described above and let $h = |T|$ be the number of terminals. Then $\mathcal{A}$ produces a Steiner tree rooted at s containing $k/8$ or more other terminals with cost-density (with respect to c) at most $c_2 \log^3 h \cdot \text{OPT}/h'$ and diameter (with respect to ℓ) at most $c_1 \log h \cdot L$, where* OPT *is the cost of an optimum k-Steiner tree with diameter bounded by L.*

Since we use the algorithm of Theorem 5.7 frequently, we refer to it in this paper as the KSLT algorithm. The main procedure in our algorithm is Procedure Jnc-Tree that tries to find a low density junction tree. This procedure works in rounds and every round is divided into two phases: the sources phase and the sinks phase. The sources phase gradually builds a tree F_s by attaching new sources into the tree at low density in iterations. After the sources phase ends a single iteration of the sinks phase takes place, in which we try to add to the tree, at low density, some of the sinks corresponding to sources that belong to F_s. If the single iteration in the sinks phase is a success then Jnc-Tree finds a partial solution of low density routing a subset of the pairs. Otherwise, part of the pairs are *temporarily* discarded and a new round of Jnc-Tree is performed restricted to undiscarded pairs. We show that eventually we find a low density junction tree before all the pairs are discarded. For a subtree F obtained by calling KSLT, $T(F)$ is the set of terminals in F. Let $\mathcal{T}'$ be the set of remaining (unrouted) pairs of the original instance.

```
Procedure Jnc-tree ($\mathcal{T}'$)

1. Let $\mathcal{T}'' \leftarrow \mathcal{T}'$ and $h' = |\mathcal{T}'|$
2. **While** $\mathcal{T}'' \neq \emptyset$ **Do**
   (a) let $s$ be an arbitrary source of a pair in $\mathcal{T}''$.        /* Phase 1: sources phase starts here*/
   (b) LowDens $\leftarrow$ true; $F_s \leftarrow s$; $k_s \leftarrow 1$; $j \leftarrow 1$        /* $F_s$ is the Steiner tree found so far */
   (c) repeat
       i. $j \leftarrow j + 1$
       ii. Find a Steiner tree $F_s^j$ rooted at $s$ by calling KSLT with parameter $k = \lceil k_s/200 \rceil$ and diameter
           bound $L = 4 \log h \cdot \text{OPT}_\ell/h'$        /* By definition $|T(F_s^j)| \geq k_s/1600$ */
       iii. If $c(F_s^j)/|T(F_s^j)| \leq 32c_2 \cdot \log^3 h \cdot \text{OPT}_c/h'$ then        /* A successful iteration */
            $F_s \leftarrow F_s \cup F_s^j$
            $k_s \leftarrow T(F_s)$        /* $k_s$ always counts the number of sources in $F_s$ */
            Contract all of $F_s^j$ into $s$
       iv. Else LowDens $\leftarrow$ False        /* A failed iteration */
   (d) until LowDens = False
   (e) Let $X(F_s)$ be the set of terminals in $F_s$ and $Y_s$ be their sinks
       /* Phase 2: sinks phase starts here*/
   (f) Obtain $F_t$ by calling KSLT with $s$ as the root, $Y_s$ as terminals, $k = \lceil k_s/100 \rceil$, and $L = 4\text{OPT}_\ell/h'$.
   (g) If $c(F_t)/|T(F_t)| \leq 16c_2 \cdot \log^3 h \cdot \text{OPT}_c/h'$ then return $E(F_s) \cup E(F_t)$ as the junction-tree and stop.
   (h) Else, discard from $\mathcal{T}''$ all the pairs whose sources are in $X(F_s)$.
```

5.3 Analysis of the Algorithm

We may assume (by duplicating nodes) that all the sources are different and all sinks are different (hence h' at the same time is the number of uncovered pairs, the number of remaining sources and the number of remaining sinks). We show that every call to Jnc-Tree finds a low density junction tree. Consider one call to Jnc-Tree with parameter $\mathcal{T}'$ (and $h' = |\mathcal{T}'|$). Assume that OPT_c and OPT_ℓ are the cost and length of the optimal solution to the original instance, respectively. Let $\mathcal{S}$ be the set of spheres (i.e. subgraphs $G_1, \ldots, G_k$) computed in the decomposition for the proof of Lemma 5.1. We call a sphere (subgraph) G_i *good* if at most a fraction $1/4$ of the source-sink pairs of G_i are discarded by the algorithm. A pair that belongs to a good sphere at the time of being considered is called a good pair and the rest are called *bad*. A source is good if it belongs to a good pair. Note that a good sphere may become bad during the course of the algorithm as some of its pairs are discarded. Accordingly, all its remaining pairs become bad. One round of Jnc-Tree is one iteration of the while loop. For every round of Jnc-Tree, trees F_s and F_t are the trees obtained at the end of the sources phase and sinks phase, respectively. We call a round of Jnc-Tree a *bad round* if the number of good sources in F_s is at most $\lfloor k_s/50 \rfloor$. That is, at most $\lfloor k_s/50 \rfloor$ of sources of F_s belong to good spheres of $\mathcal{S}$. The rest of the rounds are called *good rounds*. A good sphere $G_i \in \mathcal{S}$ that intersects F_s is called *sparse* with respect to F_s if F_s contains at most half of the original sources of G_i. A good round is a *sparse* round if among all good sources in F_s, at least half of them belong to good spheres that are sparse with respect to F_s. Other good rounds are *dense* rounds. By this definition, every round is either: (i) a bad round, or (ii) a good sparse round, or (iii) good dense round. We later show that there are no good sparse rounds at all. Only bad rounds or good dense rounds exist. We also show that if a round is good and dense, then the sinks phase cannot fail and so Jnc-Tree finds a junction tree, whose density is shown to be low. Thus, it remains to show that not all rounds of Jnc-Tree are bad. This is the first thing we prove. Note that as long as at least one source remains undiscarded, Jnc-Tree will start a new round. The only way for Jnc-Tree to fail is if all sources are discarded. The following is the main lemma we prove in this section.

Lemma 5.8 *Every call to Jnc-Tree finds a junction tree with density is at most* $O(\log^3 h \cdot \text{OPT}/h')$.

Note that Lemma 5.8 only bounds the density of every subtree returned. To get the final ratio we use Lemma 2.1. For general D, Lemma 2.1 implies that an additional factor of $O(\log D)$ is incurred.

Corollary 5.9 *The approximation ratio of the greedy algorithm is* $O(\log^3 h \cdot \log D)$.

We now end this section by presenting the proof of Lemma 5.8. First we need a series of lemmas.

Lemma 5.10 *In every call to Jnc-Tree, either the procedure finds a junction-tree and returns or there is at least one good round before all the pairs are discarded from* $\mathcal{T}''$.

Proof. Suppose by contradiction that all the rounds are bad and we continue until all the pairs are discarded from $\mathcal{T}''$. Let k_i denote the number of pairs discarded in round i. This implies that $\sum_i k_i = h'$. By property 1 of Claim 5.2, the number of sources (pairs) in $\mathcal{S}$ is at least $\lceil h'/8 \rceil$. Note that initially, all sources of $\mathcal{S}$ are good. Since we assumed each round is bad, in round i at most $\lfloor k_i/50 \rfloor$ good sources are discarded among the total of k_i discarded sources. Recall (from proof of Lemma 5.1) that $\mathcal{T}_i'$ is the number of pairs inside the sphere G_i. From each sphere $G_i \in \mathcal{S}$, the first $\mathcal{T}_i'/4$ sources selected are good and the remaining become bad (this happens when the number of undiscarded pairs in G_i goes below $\frac{3\mathcal{T}_i'}{4}$). That is, the number of good pairs that become bad is at most 3 times the number of good pairs that are discarded. Thus the total number of good pairs discarded and the number of good pairs that become bad is at most $\sum_i 4\lfloor \frac{k_i}{50} \rfloor \le \sum_i \frac{4k_i}{50} = \frac{4h'}{50} < \frac{h'}{10}$. Therefore at least $h'/8 - h'/10 = h'/40$ good pairs remain, and so the Procedure Jnc-Tree could not have removed all the sources as some good sources remain. Hence, there must be a good round. □

Lemma 5.11 *There are no good and sparse rounds.*

Proof. We proceed by contradiction. Consider the first good round and assume it is a sparse round and let q be the last successful iteration at line 2c before the single failed $(q+1)$st iteration. Therefore $F_s = \bigcup_{i=1}^{q} F_s^i$. Let

$\mathcal{S}' \subseteq \mathcal{S}$ be the collection of all the good sparse (with respect to F_s) spheres that belong to $\mathcal{S}$ and remained after all the previous (bad) rounds. If some G_i has no intersection with F_s then it is not included in $\mathcal{S}'$. Using property 2 of Claim 5.2 and since each of $G_i \in \mathcal{S}'$ intersects F_s it follows that all the nodes of $V(\mathcal{S}') = \bigcup_{G_i \in \mathcal{S}'} V(G_i)$ are within distance $2\log n \cdot \text{OPT}_\ell/h'$ of some node $u \in F_s$. Since all the spheres in $\mathcal{S}'$ are sparse, at most half the sources of the pairs in each $G_i \in \mathcal{S}'$ are actually in F_s (by the definition of a sparse round). Also, at most $\mathcal{T}'_i/4$ of the sources of G_i are discarded (or else G_i would not be good anymore). Therefore, at least $C = \sum_{G_i \in \mathcal{S}'} \frac{\mathcal{T}'_i}{4}$ sources remain (undiscarded) that do not belong to F_s. First we show that $C \geq \lceil k_s/200 \rceil$. By the definition of a good round, the number of good sources in F_s is at least $\lceil k_s/50 \rceil$. By the definition of a sparse good round at least $1/2$ of them are by sparse spheres. Hence, the number of good sources in F_s that come from sparse spheres (i.e., from spheres in $\mathcal{S}'$) is at least $\lceil k_s/100 \rceil$. Since for each $G_i \in \mathcal{S}'$, the number of sources of G_i that intersect F_s is no more than $\mathcal{T}'_i/2$, it follows that $C \geq \lceil k_s/200 \rceil$. Consider the failed iteration $q+1$. Let $E(\mathcal{S}')$ be the set of edges of the spheres in $\mathcal{S}'$ and compute the shortest path tree rooted at s (the root of F_s^q) which is obtained by taking the shortest path from s to every node in every $G_i \in \mathcal{S}'$. We obtain a tree with diameter at most $4\log n \cdot \text{OPT}_\ell/h'$ (since every node in G_i is at distance at most $2\log n \cdot \text{OPT}_\ell/h'$ from the root) and by $C \geq \lceil k_s/200 \rceil$, it contains at least $\lceil \frac{k_s}{200} \rceil$ new sources. Let H_s^{q+1} denote this tree. Thus in iteration $j = q+1$ of the repeat loop in Phase 1, there is a Steiner tree H_s^{q+1} (over $E(\mathcal{S}')$) with $\lceil \frac{k_s}{200} \rceil$ sources with diameter at most $D = 4\log n \cdot \text{OPT}_\ell/h'$. By property 4 of Claim 5.2, and since the graphs $G_i \in \mathcal{S}'$ are disjoint, the cost density of H_s^{q+1} is at most $\frac{\sum_{G_i \in \mathcal{S}'} c(G_i)}{\sum_{G_i \in \mathcal{S}'} \mathcal{T}'_i/4} \leq 32\frac{\text{OPT}_c}{h'}$. By Theorem 5.7, the density of the Steiner tree returned by KSLT algorithm is at most a factor $c_2 \log^3 h$ larger than the cost density of H_s^{q+1}. Thus the cost density of the tree F_s^{q+1} that the algorithm finds is at most $32c_2 \cdot \log^3 h \cdot \frac{\text{OPT}_c}{h'}$. Hence, the cost density of F_s^{q+1} is no larger than $32c_2 \cdot \log^3 h \cdot \frac{\text{OPT}_c}{h'}$. Thus the round should not have failed. □

Lemma 5.12 *If the round is good and dense, the sinks phase finds a low density tree and so Jnc-Tree finds a partial solution.*

Proof. If a round is good, there are at least $\lceil k_s/50 \rceil$ good sources in F_s. If it is a good and dense round then at least $\lceil k_s/100 \rceil$ good sources of F_s belong to dense good spheres. Let H be the set of these good sources (good sources in dense spheres). Define $\mathcal{S}' \subseteq \mathcal{S}$ to be the set of good dense spheres that intersect F_s. For every $s_i \in H$, its distance to t_i in $E(\mathcal{S}')$ is at most $2\text{OPT}_\ell/h'$ (by property 3 of Claim 5.2). Thus, this is also a bound on the distance from the root of F_s^q (i.e. s) to t_i. Hence, after $E(\mathcal{S}')$ is added, the shortest path tree from s to all the sinks of $s_i \in H$ has radius $2\text{OPT}_\ell/h'$. This gives a tree with diameter at most $4\text{OPT}_\ell/h'$ which is the appropriate bound. The cost density of this tree is at most $\sum_{G_i \in \mathcal{S}'} c(G_i)/|H|$. Since all $G_i \in \mathcal{S}'$ are dense, $\sum_{G_i \in \mathcal{S}'} \mathcal{T}'_i/2 \leq |H|$. This implies that $\frac{\sum_{G_i \in \mathcal{S}'} c(S_i)}{|H|} \leq \frac{\sum_{G_i \in \mathcal{S}'} c(G_i)}{\sum_{G_i \in \mathcal{S}'} \mathcal{T}'_i/2} \leq 16\text{OPT}_c/h'$, where the last inequality follows form property 4 of Claim 5.2. Therefore, there is a Steiner tree containing s and the sinks of H with diameter bound $4\text{OPT}_\ell/h'$ and cost density at most $16\text{OPT}_c/h'$. By Theorem 5.7, the density of the returned tree is bounded by $16c_2 \cdot \log^3 h \cdot \text{OPT}_C/h'$ which implies that the round is good. □

Proof of Lemma 5.8. By Lemma 5.10, before Jnc-Tree discards all sources, there must be at least one good round. By Lemma 5.11, the good round must be dense. By Lemma 5.12 such a round must succeed. Thus the procedure always finds a junction tree. Now we bound its density.

In Phase 1, the cost density of F_s is at most $O(\log^3 h \cdot \text{OPT}_c/h')$. This is explained as follows. Since every new tree added to F_s has density at most $O(\log^3 h \cdot \text{OPT}_c/h')$ this bounds the density of F_s as well.

However, note the following difference: we know that the cost over the number of sources is "low". But the number k_s of sources in F_s can be different from the number of pairs covered. However, the number of pairs covered is at least $(k_s/100)/8 = k_s/800$ (see Theorem 5.7). Thus the cost density of F_t with respect to covered pairs is bounded by $800 \cdot O(\log^3 h \cdot \text{OPT}_c/h') = O(\log^3 h \cdot \text{OPT}_c)/h'$.

Now we bound the length density. First consider Phase 1 (sources phase). By the property of Theorem 5.7, the diameter of each Steiner tree F_s^i found in each iteration i is at most $c_1 \cdot \log hL = 4c_1 \cdot \log^2 h \cdot \text{OPT}_\ell/h'$. Thus the total diameter of F_s, denoted by r_s, is at most $r_s \leq 4c_1 \cdot q \cdot \log^2 h \cdot \text{OPT}_\ell/h'$, where q is the last successful iteration. Since in every iteration of the repeat loop, the number of new sources covered is at least $(k_s/200)/8$ (see Theorem 5.7 and Line 2(c)ii in Jnc-tree) the number of sources in F_s is multiplied at least by $1601/1600$ at every iteration. Thus the number of iterations (and therefore q) is in $O(\log h)$. Thus $r_s = O(\log^3 h \cdot \text{OPT}_\ell/h')$. The diameter of F_t is at most $O(\log h \cdot \text{OPT}_\ell/h')$ by the bound L passed to KSLT in Phase 2. In total the diameter is $O(\log^3 h \cdot \text{OPT}_\ell/h')$. Hence, if we cover q pairs using F_s and F_t then the length density is at most $q \cdot O(\log^3 h \cdot \text{OPT}_\ell/h')/q$ which is $O(\log^3 h \cdot \text{OPT}_\ell/h')$. □

Greedy approximation for NMC: The greedy approximation algorithm presented in this section can be adapted to work for NMC. For that we need a node-weighted version of Theorem 5.7. In node-weighted shallow-light k-Steiner tree problem, denoted by NKSLT, we are given a graph $G(V, E)$ with node cost function $c : V \to \mathbb{R}^+$ and length function $\ell : V \to \mathbb{R}^+$, a collection T of terminals containing a root s, a number k, and a diameter bound L. The goal is to find a minimum cost (w.r.t. c) s-rooted k-Steiner tree that has diameter (w.r.t. ℓ) at most L.

Lemma 5.13 *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 gives an $O(\log^3 h \log D)$-approximation for NMC. We briefly sketch the the ideas for the proof of Lemma 5.13. The algorithm borrows ideas from the algorithm of [20] for (edge-weighted) shallow-light k-Steiner trees (Theorem 5.7) and [24] for node-weighted Steiner tree. Here we describe the similarities and differences. The algorithm for Theorem 5.7 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 5.13 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.

6 Discussion and Future Work

Table 1 summarizes the known bounds for various versions of the buy-at-bulk problem. For a single problem, namely, NSS we have a $\Theta(\log n)$-approximability threshold (namely a matching upper and lower bound for the approximation). For the other versions there is a gap between the upper and lower bounds on the approximation ratio. This gap is particularly large for NMC; $O(\log^4 h)$ vs $\Omega(\log^{\frac{1}{2}-\epsilon})$. To improve the upper bound, one needs to exploit the interaction between the algorithm for computing the minimum density junction tree and the proof of the existence of low density trees. We believe that there exists an $O(\log h)$-approximation for den-SS and den-NSS. For the uniform case it may be that the shallow-light tree theorem (Theorem 5.7) can be improved yielding an improved result for MC with polynomial demands.

A slight improvement is given in [26]. In [26] an $O(\log^3 h)$ ratio algorithm is given for the case of polynomial demands, using better LP techniques. See http://crab.rutgers.edu/∼guyk/pub.html. Unfortunately the algorithm does not work for general demands and leaves this question open.

A related question is to obtain a bound on the integrality gap of an LP formulation for NMC and MC. Such a formulation is a straightforward extension of the formulation for the single-source problem from [11]. Although we believe that a poly-logarithmic upper bound can be established on the integrality gap of this formulation, current techniques do not seem adequate and proving some polylogarithmic integrality gap remains an interesting open problem. We also observe, via a connection shown in [21], that a poly-logarithmic approximation for k-MC would imply a poly-logarithmic approximation for the k-densest subgraph problem which is a major open problem.

Acknowledgments: The second author would like to thank Kamal Jain and Kunal Talwar for some initial discussions on the MC problem. We are grateful to Harald Räcke for pointing out the idea behind Lemma 4.1 which improved our earlier version from [8]. The last author also thanks school of Computer Science at Institute for Studies in Theoretical Physics and Mathematics in Tehran as part of this work was done while visiting there.

References

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

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

[3] M. Andrews and L. Zhang. The access network design problem. *Algorithmica*, 197–215, 2002. Preliminary version in *Proc. of IEEE FOCS*, 1998.

[4] A. Agrawal, P. Klein and R. Ravi. When Trees Collide: An Approximation Algorithm for the Generalized Steiner Problem on Networks. SIAM J. Comput. 24(3): 440-456 (1995)

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

[6] Y. Bartal. On approximating arbitrary metrics by tree metrics. *Proc. of ACM STOC*, 161–168, 1997.

[7] 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.

[8] C. Chekuri, M. Hajiaghayi, G. Kortsarz, and M. Salavatipour. Approximation algorithms for non-uniform buy-at-bulk network design problems. *Proc. of IEEE FOCS*, 677–686, 2006.

[9] C. Chekuri, M. Hajiaghayi, G. Kortsarz, and M. Salavatipour. Approximation algorithms for node-weighted buy-at-bulk network design. *Proc. of ACM-SIAM SODA*, 1265–1274, 2007.

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

[11] C. Chekuri, S. Khanna and J. Naor. A Deterministic Approximation Algorithm for the Cost-Distance Problem. Short paper in *Proc. of ACM-SIAM SODA*, 232–233, 2001.

[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. *JCSS*, 69:485–497, 2004. Preliminary version in *Proc. of ACM STOC*, 2003.

[13] J. Fakcharoenphol, S. Rao, and K. Talwar. A tight bound on approximating arbitrary metrics by tree metrics.

[14] A. Gupta, A. Kumar, M. Pál, 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.

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

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

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

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

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

[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] M. Hajiaghayi, K. Jain. The Prize-Collecting Generalized Steiner Tree Problem via a new approach of Primal-Dual Schema. *Proc. of ACM-SIAM SODA*, 631–640, 2006.

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

[23] D. S. Johnson. Approximation algorithms for combinatorial problems. *JCSS*, Vol 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. Kumar, A. Gupta, and T. Roughgarden. A constant-factor approximation algorithm for the multicommodity rent-or-buy problem. *Proc. of IEEE FOCS*, 333–342, 2002.

[26] G. Kortsarz and Z. Nutov. Approximating some network design problems with vertex costs.

[27] M. Marathe, R. Ravi, R. Sundaram, S. S. Ravi, D. Rosenkrantz, H. B. Hunt. Bicriteria network design. *Journal of Algorithms*, 28(1):142–171, 1998.

[28] A. Meyerson, K. Munagala, and S. Plotkin. Cost-Distance: Two Metric Network Design.

[29] Ran Raz, Shmuel Safra. A Sub-Constant Error-Probability Low-Degree Test, and a Sub-Constant Error-Probability PCP Characterization of NP. STOC, 475-484, 1997.

[30] G. Robins and A. Zelikovsky. Tighter Bounds for Graph Steiner Tree Approximation. SIAM J. Discrete Math. 19(1): 122-134 (2005)

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

[32] F. S. Salman, J. Cheriyan, R. Ravi, and S. Subramanian. Buy at bulk network design: Approximating the single-sink edge installation problem. *Proc. of ACM-SIAM SODA*, 619–628, 1997.

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

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

A Proof of Lemma 3.2

Recall that Lemma 3.2 states that for any instance of NSS there is a spider of density at most OPT_{LP}/h. The proof is similar in spirit to that of Guha et al. [17] 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. There are two types of variables; for each terminal t there is a variable $y(t)$ and for each node $v \in V$ and terminal t there is a variable $z_t(v)$. Recall that $\mathcal{P}_t$ is the set of all paths from t to the root r.

$$\begin{array}{llcll}
\text{DP-NSS} & \max & \sum_{t\in T} y(t) & & \\
\text{subject to:} & \sum_{t\in T} z_t(v) & \leq & c(v) & v \in V \\
 & y(t) - \sum_{v\in p} z_t(v) & \leq & \delta(t)\cdot \ell(p) & p \in \mathcal{P}_t,\ t \in T \\
 & y(t), z_t(v) & \geq & 0 & v \in V, t \in T
\end{array}$$

We create a solution y', z' to DP-NSS as follows. Recall that $d_t(v) = \min_{p\in P_{tv}}(c(p) + \delta(t)\ell(p))$ is the shortest node weighted distance from t to v where each node u has weight $c(u) + \delta(t)\ell(u)$. 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 show that if y', z' is not feasible then there exists a spider of density strictly less than R. We also give a more intuitive explanation for z'.

For every terminal t we define a ball B_t of radius R with the center at t using the distance function d_t. This ball contains some nodes fully and some nodes partially. Since d_t depends on two separate node weight functions, one needs a more careful definition of when a node is contained partially in the ball B_t. For a node v and terminal t, we define two quantities $\gamma_t(v)$ and $\sigma_t(v)$ which take values in $[0, 1]$. We interpret them as follows; $\gamma_t(v)$ is the fraction of *cost* of v and $\sigma_t(v)$ is the fraction of *length* of v that belongs to ball of t. We maintain the property that when $\gamma_t(v) > 0$ we have $\sigma_t(v) = 1$, in other words we give preference to the length first and then the cost. Formally we set $\gamma_t(v)$ and $\sigma_t(v)$ as follows. We assume for simplicity that $\ell(v)$ and $c(v)$ are non-negative.

- If $d_t(v) \leq R$ then $\gamma_t(v) = \sigma_t(v) = 1$.
- If $R < d_t(v) \leq R + c(v)$ then $\gamma_t(v) = (R - d_t(v) + c(v))/c(v)$ and $\sigma_t(v) = 1$.
- If $R + c(v) < d_t(v) < R + c(v) + \delta(t)\ell(v)$ then $\gamma_t(v) = 0$ and $\sigma_t(v) = (R - d_t(v) + c(v) + \delta(t)\ell(v))/(\delta(t)\ell(v))$.
- If $R + c(v) + \delta(t)\ell(v) \leq d_t(v)$ then $\gamma_t(v) = \sigma_t(v) = 0$.

Note that by the above definitions, it is guaranteed that $\gamma_t(v), \sigma_t(v) \in [0, 1]$ and if $\sigma_t(v) < 1$ then $\gamma_t(v) = 0$ and conversely if $\gamma_t(v) > 0$ then $\sigma_t(v) = 1$. Thus nodes with $\gamma_t(v) = 1$ are completely contained in the ball B_t and nodes with $\sigma_t(v) = 0$ are outside the ball. The rest are partially inside the ball. The high level intuition for the proof is that the balls B_t for different t should be disjoint for otherwise a node in the intersection of two balls B_t and $B_{t'}$ can be used to find a spider that connects t and t' of density strictly less than R. However the formal proof becomes technical to take care of the case of nodes that are partially in more than one ball.

The claim below can be shown by simple case analysis using the definitions.

Claim A.1 *$z'_t(v) = \gamma_t(v) \cdot c(v)$ and $z'_t(v) < c(v)$ if and only if $d_t(v) \geq R + (1 - \gamma_t(v)) \cdot c(v) + (1 - \sigma_t(v)) \cdot \delta(t)\ell(v)$.*

Let P_{tv} be the set of all paths between a terminal t and a node v.

Claim A.2 *For any terminal t and node v and any path $p \in P_{tv}$, if $z'_t(p) + \delta(t)\ell(p) < R$ then $d_t(v) < R$.*

Proof. Consider a terminal t and a node v and a path $p \in P_{tv}$ such that $z'_t(p) + \delta(t)\ell(p) < R$. We claim that every node u in p satisfies the property that $z'_t(u) = c(u)$. Suppose not. Let w be the first node in the path starting from t such that $z'_t(w) < c(w)$. From Claim A.1 we have that

$$d_t(w) \geq R + (1 - \gamma_t(w))c(w) + (1 - \sigma_t(w))\ell(w) \geq R + (1 - \gamma_t(w))c(w).$$

Consider the subpath q of p from t to w and let q' be the path q with w omitted. Note that for every node u on q', $z'_t(u) = c(u)$. We have

$$\begin{aligned}
z'_t(p) + \delta(t)\ell(p) &\geq z'_t(q) + \delta(t)\ell(q) \\
&= z'_t(q') + \delta(t)\ell(q') + z'_t(w) + \delta(t)\ell(w) \\
&= c(q') + \delta(t)\ell(q') + z'_t(w) + \delta(t)\ell(w) \\
&= c(q') + \delta(t)\ell(q') + c(w) + \delta(t)\ell(w) - (c(w) - z'_t(w)) \\
&= c(q) + \delta(t)\ell(q) - (1 - \gamma_t(w))c(w) \\
&\geq d_t(w) - (1 - \gamma_t(w))c(w) \\
&\geq R,
\end{aligned}$$

which contradicts our assumption that $z'_t(p) + \delta(t)\ell(p) < R$.

Therefore all nodes u in p have the property that $z'_t(u) = c(u)$. Then $z'_t(p) + \delta(t)\ell(p) = c(p) + \delta(t)\ell(p) < R$. Since $d_t(v) \leq c(p) + \delta(t)\ell(p)$ for any $p \in P_{tv}$ we have $d_t(v) < R$ as desired. $\square$

Now we are ready to prove that y', z' is feasible for DP-NSS.

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 A.2, for any path $p \in \mathcal{P}_t$, $z'_t(p) + \delta(t)\ell(p) \geq R$ which proves that the second set of constraints in DP-NSS is satisfied.

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 $t \in T_s$. Thus the first set of constraints are also satisfied.