Electronic Colloquium on Computational Complexity, Report No. 7 (2006)

Approximating Buy-at-Bulk k-Steiner trees

MohammadTaghi Hajiaghayi [*] Guy Kortsarz[†]
Mohammad R. Salavatipour[‡]

Abstract

In the buy-at-bulk k-Steiner tree (or rent-or-buy k-Steiner tree) problem we are given a graph $G(V,E)$ with a set of terminals $T \subseteq V$ including a particular vertex s called the root, and an integer $k \leq |T|$. There are two cost functions on the edges of G, a buy cost $b : E \longrightarrow \mathbb{R}^+$ and a rent cost $r : E \longrightarrow \mathbb{R}^+$. The goal is to find a subtree H of G rooted at s with at least k terminals so that the cost $\sum_{e\in H} b(e) + \sum_{t\in T-s} dist(t,s)$ is minimize, where $dist(t,s)$ is the distance from t to s in H with respect to the r cost. Our main result is an $O(\log^5 n)$-approximation for the buy-at-bulk k-Steiner tree problem. To achieve this we also design an approximation algorithm for bicriteria k-Steiner tree. In the bicriteria k-Steiner tree problem we are given a graph G with edge costs $b(e)$ and distance costs $r(e)$ over the edges, and an integer k. Our goal is to find a minimum cost (under b-cost) k-Steiner tree such that the diameter under r-cost is at most some given bound D. An (α, β)-approximation finds a subgraph of diameter at most $\alpha \cdot D$ (with respect to r) and cost with respect to b of at most $\beta \cdot opt$ where opt is the minimum cost of any solution with diameter at most D. Marathe et al [25] gave an $(O(\log n), O(\log n))$-approximation algorithm for the bicriteria Steiner tree problem. Their algorithm does not extend to the bicriteria k-Steiner tree problem. Our algorithm for the buy-at-bulk k-Steiner tree problem relies on an $(O(\log^2 n), O(\log^4 n))$-approximation algorithm we develop for the (shallow-light) bicriteria k-Steiner tree problem, which is of independent interest. Indeed, this is also one of the main tools we use to obtain the first polylogarithmic approximation algorithm for non-uniform multicommodity buy-at-bulk [21].

1 Introduction

Recently, there has been growing interest in network design problems with two cost functions (instead of one) partially due to practical applications. For most of such problems so far there is no general tool that would facilitate their solutions such as Bartal's result of embedding a metric into probabilistic trees. In this paper, we study the *buy-at-bulk k-Steiner tree* (or *rent-*

[*]Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology. This research was supported in part by IPM under grant number CS1383-2-02. E-mail: `hajiagha@mit.edu`.
[†]Department of Computer Science, Rutgers University-Camden. E-mail: `guyk@crab.rutgers.edu`.
[‡]Department of Computing Science, University of Alberta. E-mail: `mreza@cs.ualberta.ca`. Supported by NSERC and a faculty start-up grant.

ISSN 1433-8092

or-buy k-Steiner tree) problem[1]. In this problem we are given an undirected graph $G(V, E)$ with a terminal set $T \subseteq V$, a specific vertex $s \in T$ called *the root*, and an integer $k \leq |V| = n$. We also have two (independent) cost functions on the edges of G: buy cost $b : E \longrightarrow \mathbb{R}^+$ and rent cost $r : E \longrightarrow \mathbb{R}^+$. Usually, when r and b of different edges can be different the term *non-uniform* is used. We omit this term and note that all the variants studied here are non-uniform. Our goal is to find a Steiner tree H spanning at least k vertices of T including the root which minimizes the following:

$$\sum_{e \in H} b(e) + \sum_{t \in T-s} L(t, s), \tag{1}$$

where $L(t, s) = \sum_{e \in P(t,s)} r(e)$ with $P(t, s)$ being the unique path from t to s in H.

Buy-at-bulk network optimization problems with two cost functions, have been extensively studied, sometimes under different names such as cost-distance (where one function defines buy cost and another function defines length) and rent and buy problems. These problems have practical importance. Consider for example the case that every edge can either be bought or rented (but not both). The chosen edges are required to provide bandwidth to satisfy some multicommodity demands. As we shall see later, this model is equivalent to the above buy-at-bulk model (in which every edge has both rent and buy costs). The rent or buy (but not both) version of the problem captures the practical setting where it is needed to decide whether to buy an edge and pay $b(e)$ once (typically $b(e)$ is relatively large) or to rent the edge e. If the user decides to rent the edge, then the pay is $r(e)$ per every commodity that goes via e. For its practical importance, this problem has been studied through a long line of papers in the operation research and computer science communities after the problem was introduced by Salman et al. [29] (see e.g. [2, 8, 16, 17, 18, 19, 24, 26, 30]). We will show in Subsection 1.3 that this model is in fact equivalent to the rent and buy model described earlier.

Problems that require to satisfy at least k of the demands and often involve (two) cost functions also attracted considerable attention and so far there seems to be no general unifying technique for these problems (see e.g. [4, 7, 10, 14, 20, 23, 27]). An example for such a technique is the Lagrangian relaxation technique together with the primal-dual method that was used to derive a constant-factor approximation algorithms, e.g., for k-MST or k-Steiner tree [10]. However, this technique does not seem to work when two cost functions are involved.

We give the first approximation algorithm for the buy-at-bulk k-Steiner tree problem, of ratio $O(\log^5 n)$. The approximation algorithm we develop for the buy-at-bulk k-Steiner tree problem uses an approximation for a variant of the shallow-light network design problem that is of independent interest. A graph $G(V, E)$ and a collection $T \subseteq V$ of terminals are given in addition to cost and length functions $b, r : E \longrightarrow \mathbb{R}^+$ and two bounds, a cost bound B and a length bound D. The cost of a spanning subtree $H(V, E')$ is $b(E') = \sum_{e \in E'} b(e)$. For a path P, $L(P) = \sum_{e \in P} r(e)$. The distance between u, v in H is $\text{dist}_H(u, v) = L(P_{u,v})$ so that $P_{u,v}$ is the unique path between u, v in H. The diameter of H is $\text{diam}(H) = \max_{u,v} \text{dist}_H(u, v)$. Assuming a spanning subtree $H(V, E')$ with cost at most B and diameter at most D exists, the shallow-light spanning tree problem is to find H. The more general shallow-light k-Steiner

[1]Both terms "buy-at-bulk" and "rent-or-buy" are used interchangeably in the literature. We mainly use the former term in this paper.

tree problem requires to select for an input k a tree spanning k nodes that meets the diameter and costs bounds D and B respectively. Even the shallow-light spanning tree ($k = n$) special case is NP-hard and also NP-hard to approximate within a factor better than $c \log n$ for some universal constant c [5]. Thus we focus on an approximation algorithm. An (α, β) bi-criteria approximation algorithm for the shallow-light k-Steiner tree problem is an algorithm that delivers a tree H' with k terminals (vertices in T) whose diameter is at most $\alpha \cdot D$, and whose cost is at most β times the cost of a D-diameter minimum cost tree.

The constraint that only $k < n$ nodes have to be picked makes this problem considerably harder than the usual shallow-light spanning tree problem, namely, the $k = n$ case. In [25] the case $k = n$ is studied and an $(O(\log n), O(\log n))$-bicriteria approximation is given. However no approximation was known for general k prior to this paper.

1.1 Related work

In the buy-at-bulk multicommodity problem we are given p source-sink pairs, $\{s_i, t_i\}_{i=1}^{p}$. A subset E' of the edges is feasible if for every i, an s_i to t_i path exists in $G' = (V, E')$, namely, s_i, t_i belong to the same connected component in G'. The cost of E' is $\sum_{e \in E'} b(e) + \sum_i \text{dist}_{G'}(s_i, t_i)$ where the distance is with respect to r, and it is required to find a minimum cost feasible E'. If we are also given an integer $k \leq p$ and must find a solution that connects k (out of p) s_i, t_i pairs then we have the buy-at-bulk k-multicommodity problem. It is easy to see that the buy-at-bulk Steiner (but not k-Steiner) tree problem is a special case of the buy-at-bulk multicommodity problem in which all the sinks are at a single vertex (namely the root). The buy-at-bulk k-Steiner tree problem is a special case of the buy-at-bulk k-multicommodity problem. However, it is shown in [20] that if the buy-at-bulk k-multicommodity problem admits a polylogarithmic ratio approximation then so does the dense-k-subgraph problem (see [13]). For a long time now (almost 10 years) the best known approximation for the dense k-subgraph problem is $O(n^{1/3-\varepsilon})$ for some positive $\varepsilon > 0$ [13], and it is widely believed that the dense k-subgraph problem admits no polylogarithmic ratio approximation. If indeed, the dense k-subgraph problem admits no polylogarithmic approximation, then the result in our paper shows that the case of single source (but many sinks) namely, buy-at-bulk k-Steiner tree is provably easier to approximate than the general case of arbitrary source-sink pairs.

In the uniform version of the buy-at-bulk multicommodity problem all the buy values along edges are equal. The best approximation known for the uniform case is $O(\log n)$ due to the results of Awerbuch and Azar [3], Bartal [6] and Fakcharoenphol et al. [12]. Kumar et al. [24] and Gupta et al. [18] present constant factor approximation for a *rent-or-buy* model (see Section 1.3) in which the cost of buying each edge is equal to M times the cost of renting the edge (per unit length) for a fixed M. The single sink uniform version also admits constant-factor approximation algorithms [17, 19]. For a bit more general setting of the single sink uniform version in which the cost functions are unknown concave functions, Goel and Estrin [15] give an $O(\log n)$ approximation algorithm.

Meyerson et al. [26] study the buy-at-bulk Steiner tree or equivalently, the non-uniform single sink buy-at-bulk multicommodity problem for which they give a randomized $O(\log n)$-approximation that was derandomized by Chekuri, Khanna, and Naor [9] via an LP formulation. Note that none of these algorithms yield any polylogarithmic ratio approximation for the

k-Steiner tree case. For the most general case, the best known ratio for the non-uniform buy-at-bulk multicommodity problem is $exp(O(\sqrt{\log n \log\log n}))$ by Charikar and Karagiozova [8]. Recently [21], we have improved this result to a polylogarithmic factor approximation using the results of this paper.

On the lower bound side, Andrews [1] showed that unless NP $\subseteq$ ZPTIME($n^{\text{polylog}\, n}$) the buy-at-bulk multicommodity problem has no $O(\log^{1/2-\varepsilon} n)$-approximation for any $\varepsilon > 0$. Under the same assumption, the uniform variant admits no $O(\log^{1/4-\varepsilon} n)$-approximation for any constant $\varepsilon > 0$. For the single sink case, Chuzhoy et al. [11] show that the problem cannot be approximated better than $\Omega(\log\log n)$ unless NP $\subseteq$ DTIME($n^{\log\log\log n}$).

Last but not least, the buy-at-bulk k-Steiner tree problem generalizes the classic Steiner tree, k-MST, and more generally k-Steiner tree problems when the rent cost is zero. For these problems, there are 1.55-approximation, 2-approximation and 4-approximation algorithms, respectively (see [28, 14, 10] for a complete literature review of these problems). As we mentioned above, the buy-at-bulk Steiner tree problem first was studied by Meyerson et al. [26], but we are not aware of any result on buy-at-bulk k-MST or buy-at-bulk k-Steiner tree.

1.2 Our results

We give the first approximation algorithm for the buy-at-bulk k-Steiner tree problem:

Theorem 1.1 *There is a polynomial time $O(\log^5 n)$-approximation for the buy-at-bulk k-Steiner tree problem*

This is achieved by first proving the following bicriteria approximation theorem.

Theorem 1.2 *The shallow-light k-Steiner subtree problem admits an $(O(\log^2 n), O(\log^4 n))$ bicriteria approximation that runs in polynomial time*

It is worth mentioning that Theorem 1.2 is one of the main tools we use to obtain the first polylogarithmic approximation algorithm for the non-uniform multicommodity buy-at-bulk problem [21].

1.3 Equivalent models

There are three different models for the buy-at-bulk k-Steiner tree problem. This holds true also for the other versions of the problem like the buy-at-bulk multicommodity problem. Below we describe these three models and show that in fact they are all equivalent.

Model A. The *unique cost* model: Every edge e of G is given with either a buy cost $b(e)$ or a rent cost $r(e)$. For buy edges we have to pay $b(e)$ to use them. For rent edges we have to pay $r(e) \cdot f(e)$ where $f(e)$ is the amount of demand routed over that edge, i.e., the number of terminals $t \in T - s$ for which e lies on the unique s, t-path in the Steiner tree solution.

Model B. The *rent or buy* model: Every edge e is given with both a buy cost $b(e)$ and a rent cost $r(e)$. For every edge in the solution, we have to decide whether to buy e and pay $b(e)$ or rent it and pay $r(e) \cdot f(e)$. However, it is not possible to both buy and rent the edge.

Model C. The *rent and buy* or *cost-distance* model: This is the model we defined earlier, i.e. every edge is given with both a buy cost $b(e)$ and a rent cost $r(e)$. The cost we pay for every edge in the solution is $b(e) + r(e) \cdot f(e)$. This model first has been considered by Meyerson et al. [26].

The following theorem shows that all these three models are in fact equivalent. We will be working with Model C in the rest of the paper.

Theorem 1.3 *All the three models A, B, and C are equivalent.*

Proof: We show how each model can be formulated by the other models. Here, we write, e.g., $C \longrightarrow A$ short for "it is possible to model Model A via Model C".

1. $C \longrightarrow A$: We mimic model A by model C as follows. We will have the same set of vertices as in model A. An edge that does not have a buy cost (in A) is given 0 buy cost in C, and otherwise, it is given 0 rent cost. Note that by definition of Model C zero rent or buy costs do not affect the overall cost and so can be ignored. This is the same affect as in model A.

2. $B \longrightarrow C$: An edge $e = (v, u)$ in model C is replaced by two edges $e_1 = (v, w_e)$, $e_2 = (w_e, u)$ (in model B) for some new and specific for e vertex w_e. The buy cost of (v, w_e) is $b(e)$ and its rent cost is ∞. The rent cost of (w_e, u) is $r(e)$ and its buy cost is ∞. The new w_e nodes are not terminals. The edges e_1 cannot be rented and the edge e_2 cannot be bought. Observe that for the two serial edge e_1, e_2 to be used, a solution both has to pay $b(e)$ buy cost and $r(e)$ rent cost for using these edges. This mimics model C.

3. $A \longrightarrow B$: An edge $e = (u, v)$ in B is replaced by by two parallel edges e_1, e_2 between u, v (it can be replaced by paths of length 2 if parallel edges are needed to be avoided). The edge e_1 will have rent cost $r(e)$ and the edge e_2 will have buy cost $b(e)$. Clearly, if e_2 is bought then there is no need to rent e_1. Otherwise, e_1 needs to be rented, unless the (u, v) is not used. This is equivalent to choosing exactly either to rent or to buy e (but not both).

■

2 The algorithm

2.1 Reducing the problem to a shallow-light k-Steiner tree problem

We show how to reduce the buy-at-bulk k-Steiner tree problem to a shallow-light k-Steiner tree problem with a logarithmic loss in the value of the optimal solution. A bicriteria network design problem [25] (A, B, S) is defined by identifying two objective functions, A and B, and specifying a membership requirement in a class of subgraphs S. Typically, there is a budget constraint on the first objective and we seek to minimize the second objective function. This way, the (diameter, cost, k-Steiner tree) problem is naturally defined as follows: we are given an undirected graph $G(V, E)$ with terminal set T, an integer $k \leq |T|$, diameter bound D, and two cost functions $b : E \longrightarrow \mathbb{R}^+$ and $r : E \longrightarrow \mathbb{R}^+$ on the edges. Our goal is to find a minimum b-cost (i.e. minimizing the cost under the b function) Steiner tree with k terminals in G such that the diameter of the tree under the r-cost is at most D. We can assume that a particular terminal $s \in T$, called the root belongs to the solution (we can simply guess this node s). Therefore, we are solving the rooted shallow-light k-Steiner tree.

We say an algorithm is an (α, β)-approximation for an (A, B, S)-bicriteria problem if in the solution produced the first objective (A) value is within factor at most α of the budget and the second objective (B) value is at most β times the minimum for any solution that is within the budget on A. Marathe et al. [25] gave an $(O(\log n), O(\log n))$-approximation for the (diameter, cost, Spanning tree) problem. We will give an $(O(\log^2 n), O(\log^4 n))$-approximation for the (rooted) k-Steiner tree problem. Then we show how we can use this as a subroutine to obtain an $O(\log^5 n)$-approximation for the (rooted) buy-at-bulk k-Steiner tree problem.

First note that by doing a binary search:

Observation 2.1 *We can assume we know the value of an optimum solution. Let OPT denote this value.*

Lemma 2.2 *If there is an (α, β)-approximation for the shallow-light rooted (diameter, cost, k-Steiner tree) problem then we have an $O((\alpha + \beta) \log n)$-approximation for (rooted) buy-at-bulk k-Steiner tree problem*

Proof: The algorithm for this lemma is inspired by the algorithm of Awerbuch et al. [4] for (standard) k-MST. Consider the input graph $G(V, E)$ for the buy-at-bulk k-Steiner tree problem. By Observation 2.1 we can assume we know OPT (the value of optimum solution). We mark every vertex with r-distance larger than OPT to s as "to be ignored". Clearly these vertices cannot be part of any optimal solution. Then, while $k > 0$ we do the following steps:

1. Run the (α, β)-approximation algorithm **A** for the minimum b-cost $\frac{k}{2}$-Steiner tree with diameter (under r-cost) bounded by $D = \frac{4OPT}{k}$.
2. Mark all the terminals (except the root) of the solution of **A** as Steiner nodes.
3. Decrease k by the number of new terminals found in this stage (i.e. $k/2$).

Since the root belongs to all the (sub)trees found in each iteration of the while loop, at the end we will get a connected graph (tree) which spans k terminals. Now we upper bound the cost of the solution. Note that the number of iterations of this algorithm is $O(\log k)$.

Consider an optimal solution H^* for buy-at-bulk k-Steiner tree instance and iteratively delete every leaf of H^* with r-distance to s (the root) larger than $\frac{2OPT}{k}$. We delete at most $\frac{k}{2}$ terminals. Otherwise, more than $k/2$ terminals have rent distance at least $2OPT/k$ and this is a contradiction as the total cost is more than OPT. So we are left with a tree containing $\frac{k}{2}$ terminals in which the r-diameter is at most $\frac{4OPT}{k}$. Thus the first iteration of the while loop finds a $\frac{k}{2}$-Steiner tree H_1 with b-cost at most $\beta \cdot OPT$ and r-diameter at most $\frac{4\alpha \cdot OPT}{k}$. Similarly, if we delete from (original) H^* every leaf with r-distance to s larger than $\frac{4OPT}{k}$ then we delete at most $\frac{k}{4}$ terminals. Therefore, there are still at least $\frac{k}{4}$ vertices in $H^* - H_1$ which are at distance at most $\frac{4OPT}{k}$ from s. By a similar argument and proceeding by induction on

i (the iteration of the while loop), the algorithm finds a $\frac{k}{2^i}$-Steiner tree H_i with b-cost at most $\beta \cdot OPT$ and r-diameter at most $\frac{2\times 2^i \alpha \cdot OPT}{k}$. Overall, after at most $O(\log k)$ iterations, we have a tree with k terminals whose total cost is at most $O((\alpha+\beta)OPT\log k)$. ∎

2.2 Algorithm for shallow-light (diameter, cost, k-Steiner tree)

In this subsection we present an $(O(\log^2 n), O(\log^4 n))$-approximation for the (rooted) shallow-light k-Steiner tree problem. Our algorithm for shallow-light k-Steiner tree is inspired by the algorithms of [25] (for shallow-light Steiner tree) and [4] for the (standard) k-MST problem.

The input consists of a graph $G(V,E)$ with two edge costs b and r, D is a bound on the diameter under the r cost, $T \subseteq V$ is the set of terminals including the root s, k is the number of terminals we wish to cover, and ε is an error parameter. In this section, whenever we talk about the cost of a path or the cost of a tree we mean the cost under b-cost and whenever we say length or diameter we mean under r-cost.

First we transform the input graph G into another graph, G^c, which we call the completion of G by doing the following. For every pair of vertices $u, v \in V$ we find a $(1+\varepsilon)$-approximate minimum cost u,v-path under b-cost with length (under r-cost) at most $2D$. Let $p^*(u,v)$ denote this cost. For this, we use the FPTAS algorithm of Hassin [22] which runs in time $O(|E|(\frac{n^2}{\varepsilon}\log\frac{n}{\varepsilon}))$. We add a new edge between u and v with b-cost equal to the cost of $p^*(u,v)$ and r-cost equal to the length of $p^*(u,v)$. Later on, in any solution of G^c that uses this new edge, we can replace it with path $p^*(u,v)$ in G at no extra cost and without increasing the length (diameter). Therefore:

Lemma 2.3 *If we have a bicriteria solution of cost X and diameter Y in G^c then we can find (in polynomial time) a bicriteria solution of cost at most X and diameter at most Y in G.*

By this lemma, and since $G \subseteq G^c$, it is enough to work with graph G^c. Note that we can delete every vertex which is not connected to s by a new edge (because the r-distance of it to s is larger than D). So all the vertices are at distance at most D from s and so are at distance at most $2D$ from each other in G. Thus we can assume G^c is a complete (multi)graph.

Before presenting the algorithm, we should note that the "rooted" and "un-rooted" versions of this problem are reducible to each other at the cost of a constant factor loss in the approximation ratio. Clearly, if we can solve the rooted version we can also solve the un-rooted version by simply trying all the terminals as the root and choose the smallest solution. On the other hand, if we have an algorithm for the un-rooted version we can do the following. Delete every node $v \in G^c$ for which the cost of edge sv is larger than $(1+\varepsilon)OPT$. Solve the un-rooted problem and if the solution does not contain the root v then add the root. This is done by arbitrarily adding an edge from v to some node in T. This will increase the cost by at most $(1+\varepsilon)OPT$ and the diameter by at most $2D$. Hence, it is enough to present an approximation algorithm for "un-rooted" shallow-light k-Steiner tree.

We focus on graph G^c and give an algorithm which finds a shallow-light $\frac{k}{8}$-Steiner tree in it that has cost at most $O(\log^3 n \cdot OPT)$ and diameter at most $O(D\log n)$. Using this algorithm iteratively similarly to the algorithm of Lemma 2.2, in $O(\log n)$ rounds, we find a shallow-light

k-Steiner tree with cost at most $O(\log^4 n \cdot OPT)$ and diameter at most $O(D \log^2 n)$.

The algorithm runs in rounds and in every round we may have several iterations (of some loop). At every round we start with every terminal as a singleton connected component. Initially, every terminal is the center of its own component. In every iteration of a round we perform a *test*. Each test has one of two outcomes: "success" or "failure". If the test is a successes, we merge two connected components and go to the next iteration. A single failure in a round causes the entire round to be a failure (so we end that round). After a failed round some of the terminals are deleted, we exit the loop, and start the next round of algorithm with a new (smaller) set of terminals. As stated above we initialize again each terminal to be a component of size 1, ignoring any mergers that were done in the last failed round.

Our goal is to find a connected component (tree) containing at least $k/8$ terminals. We say that a round is *failure free* if it has no failures at all. The number of connected components is reduced by 1 by every test that ends with successes. Thus, a failure free round will eventually end with a connected component with at least $\frac{k}{8}$ terminals. Clearly, either we fail at every round, in which case the number of terminals turns eventually empty, or we will eventually have a failure free round. We later show that the first case above cannot happen.

Assume that in round i of the algorithm the number of terminals is t_i, where $t_1 = t = |T|$. At each iteration of the loop in each round i, we divide the connected components into $O(\log t_i)$ clusters, where cluster j contains the connected components whose number of terminals is between $t_i/2^{j+1}$ and $t_i/2^j$, for $j \geq 3$.

Definition 2.1 *In every iteration of round i (for every $i \geq 1$), a cluster is called* light *if the total number of terminals in the union of the connected components in that cluster is at most $\frac{t_i}{2\log t_i}$. Otherwise, it is called* heavy.

Lemma 2.4 *In every iteration of round i (for every $i \geq 1$) there are at least $\frac{t_i}{2}$ terminals in heavy clusters.*

Proof: There are at most $\log t_i$ light clusters, and therefore they have a total of at most $\frac{t_i}{2}$ terminals. The rest of the terminals must belong to heavy clusters. ∎

In any round i and any iteration of this round, we compute the light and heavy clusters. Assuming that there are at least $\frac{k}{2}$ terminals remaining in G^c, we show (in the Main Lemma) that there is a heavy cluster with at least two connected components. Then we pick such a heavy cluster arbitrarily, say cluster C_j. Assume that all the components of C_j have between p and $2p$ terminals where $p = t_i/2^{j+1}$. For every two components in C_j we consider the edge connecting their centers (recall that since we are in G^c this edge is obtained from the approximate minimum cost path with length at most $2D$ between those vertices in G). Two connected components c_a and c_b in C_j are called *reachable* if the cost of the edge connecting their centers is at most $16 \log^2 t \cdot OPT \cdot p/k$. We test to see if there is a pair of reachable connected components in C_j. If there is such a pair of components, then we merge the components by adding the edge between their centers and then charge every node in the two components by $8 \log^2 t \cdot OPT/k$. Since there are at least $2p$ vertices in c_a and c_b combined, the total charge is enough to pay for the cost of connecting the two components. We make one of the centers of c_a or c_b (arbitrarily) to be the new center of the new (merged) component and proceed to the next iteration of this round.

Otherwise, if our test fails because there are no two reachable centers in C_j (i.e. the cost of every edge between the centers of components in C_j is larger than $16\log^2 t \cdot OPT \cdot p/k$) then we delete all the centers of the connected components of C_j (which are all terminals). Assuming that C_j has x_j components, we set $t_{i+1} = t_i - x_j$, and then exit the loop and start round $i+1$. Below is the formal description of the algorithm.

1. Set the counter i (for round) to 1 and let $t_1 = t = |T|$.
2. Every terminal is a connected component by itself and is the center of that component.
3. Repeat until there is a connected component with $\frac{k}{8}$ terminals:
   (a) Compute light and heavy clusters.
   (b) Throw away (ignore) every heavy cluster which has only one connected component and pick an arbitrary heavy cluster, say C_j, which has at least two components.
   (c) If there are two components c_a and c_b in C_j such that the cost of the edge connecting their centers is at most $16\log^2 t \cdot OPT/k$ then:
   /* The test succeeded */
      i. Merge the components by adding that edge.
      ii. Select arbitrarily one of the two centers as the center of the merged component
      iii. Charge every node in the two components by $8\log^2 t \cdot OPT/k$.
      iv. Make one of the two centers the center of the new (merged) component and goto step (a).
   (d) Otherwise, /* The test failed */
      i. Delete all the centers of components of C_j and reset the charges of all nodes to 0.
      ii. Set $t_{i+1} = t_i - x_j$ where x_j is the number of components of C_j.
      iii. Set $i = i+1$, exit this loop and goto Step 2.

Lemma 2.5 *In any round $i \geq 1$, every component participates in at most $O(\log n)$ merger operations.*

Proof: Each time a component participates in a merger the number of terminals of the components it belongs to is multiplied by at least $\frac{3}{2}$. This follows as the size of the large component is at most $2p$ for some integer p and of the smaller one at least p. Therefore there are at most $O(\log n)$ (or more precisely $O(\log k)$) iterations involving that component. ■

Lemma 2.6 *In any round $i \geq 1$ of algorithm, for every component c_a that is obtained from α merge operations, the length (under r-cost) between the center of c_a and any other node in c_a is at most $2\alpha D$.*

Proof: The proof is by induction on α and noting the fact that whenever we merge two components the length of the edge we add (between the centers) is at most $2D$. ■

Corollary 2.7 *In any round $i \geq 1$, every component has diameter at most $O(D\log n)$, always.*

Proof: Follows from Lemmas 2.5 and 2.6. ■

Lemma 2.8 *In any round $i \geq 1$, every terminal is charged at most $O(\log n)$ times and the total charge of every terminal is $O(\log^3 n \cdot OPT/k)$.*

Proof: Recall that every time a terminals is charged, the number of terminals in its new (merged) cluster grows by at least a 3/2 factor. Thus, each terminal participates in at most $O(\log n)$ mergers before we find a component with $\frac{k}{8}$ terminals or before the round fails (after which the charges are all reset to zero). Furthermore, each time a terminal is charged $8 \log^2 t \cdot OPT/k$. So the total charge of every terminal at any given time is $O(\log^3 n \cdot OPT/k)$ ∎

By this lemma, if the algorithm terminates with a $\frac{k}{8}$-Steiner tree then the cost of the tree is at most $O(\log^3 n \cdot OPT)$. Also, by Corollary 2.7 the diameter is at most $O(D \log n)$. Thus we only need to argue that the algorithm does find a $\frac{k}{8}$-Steiner tree and for that we need to show that the algorithm terminates before the number of terminals goes down below $\frac{k}{8}$. Since at every failed round the number of terminals is reduced, after at most t rounds the number of terminals becomes zero unless the algorithm terminates earlier with a feasible solution. Hence, if we show that the the set of terminals can never be smaller than $\frac{k}{2}$ then it means that the algorithm terminates before we have fewer than $\frac{k}{2}$ terminals. We also need to prove that we can perform step 3(b) of algorithm (i.e. find a heavy cluster with at least two connected components). These are proved in our main lemma, below. For that, we use the following pairing lemma:

Lemma 2.9 [25] *Let T be an arbitrary tree and let $v_1, v_2, \ldots, v_{2q}$ be an even number of vertices in T. There exists a pairing of the v_i (into q pairs) so that the unique paths joining the respective pairs are edge-disjoint.*

In the following lemma, we claim some properties on terminals not previously discarded by some failed round. We fix some optimal tree OPT and use that tree for proving these claims. As some of the terminals in OPT may have been deleted by the failed rounds, the original OPT as defined *over* G does not exist any longer (the removal of deleted terminals may have destroyed that tree). Nevertheless, we can still use this original OPT to prove properties on G^c.

Lemma 2.10 (Main Lemma) *At the beginning of any round $i \geq 1$, if the number of terminals is $t_i \geq \frac{k}{2}$ then:*

1. *There is at least one heavy cluster with at least two connected components (so we can perform step 3(b) of the algorithm).*
2. *The number of terminals in round $i+1$ (if there is such a round) is at least $k/2$.*

Proof:

1) By Lemma 2.4 there are at least $\frac{t_i}{2} \geq \frac{k}{4}$ terminals in heavy clusters. Throw away every cluster with only one connected component. These components have a total of at most $\frac{k}{8} + \frac{k}{16} + \ldots < \frac{k}{4}$ terminals. Therefore, there is at least one heavy cluster with at least two components.

2) We prove that the number of remaining terminals is always at least $\frac{k}{2}$. We use OPT to refer to both an optimal solution and the cost of that optimal solution. Let k_i be the number

of terminals of OPT that are in G^c at the beginning of round i; so $k_1 = k$. Note that always $k_i \leq t_i$. Suppose at some iteration of round i and for some heavy cluster C_j chosen by the algorithm, no pair of centers are reachable to each other; so we have to delete all the centers of C_j from G^c. Assume that all the components of C_j have size between p_i and $2p_i$.

Claim 2.11 *The number of centers of components of C_j that belong to OPT is at most $k/(8p_i \log^2 t)$.*

Proof: Otherwise, using the pairing lemma (Lemma 2.9), we can pair those centers in OPT such that the paths connecting the pairs in OPT are all edge-disjoint. By averaging, there is at least one path with cost at most $16p_i \log^2 t \cdot OPT/k$ contradicting our assumption (because if there was such a path we would have merged the two components). ■

Therefore, by Claim 2.11, the number of terminals of OPT in G^c goes down by a factor of at most $1 - 1/(8p_i \log^2 t)$. On the other hand, since C_j is a heavy cluster and we have at most $2p_i$ nodes in every component of C_j, there are at least $t_i/(2\log t_i)/(2p_i) = t_i/(4p_i \log t_i)$ components in C_j. This is also a lower bound on the number of centers (terminals) that are deleted in round i. Therefore, the number of terminals in G^c goes down by a factor of at least $1 - 1/(4p_i \log t_i)$. Hence:

$$k_i\left(1 - \frac{1}{8p_i \log^2 t}\right) \leq k_{i+1} \leq t_{i+1} \leq t_i\left(1 - \frac{1}{4p_i \log t_i}\right) \leq t_i\left(1 - \frac{1}{4p_i \log t}\right)$$

We now use the following two inequalities:

$$\text{If } x \leq 1/2, \text{ then } 1 - x \geq exp(-2x) \tag{2}$$

and

$$1 - x \leq exp(-x) \tag{3}$$

Using Inequality (2) and since $1/(8p_i \log^2 t) < 1/2$, it follows that $1 - 1/(8p_i \log^2 t) \geq e^{-1/(4p_i \log^2 t)}$. On the other hand from Inequality (3): $(1 - \frac{1}{4p_i \log t}) \leq e^{-1/(4p_i \log t)}$. Thus

$$\begin{aligned} k \cdot exp\left(-\sum_{\ell=1}^{i} \frac{1}{4p_\ell \log^2 t}\right) &\leq k\prod_{\ell=1}^{i}\left(1 - \frac{1}{8p_\ell \log^2 t}\right) \leq k_{i+1} \leq t_{i+1} \\ &\leq t\prod_{\ell=1}^{i}\left(1 - \frac{1}{4p_\ell \log t}\right) \leq t \cdot exp\left(-\sum_{\ell=1}^{i} \frac{1}{4p_\ell \log t}\right). \end{aligned}$$

Note that both sequences t_i and k_i are decreasing but at different rates and t_i is lower bounded by k_i.

Note that $\sum_{\ell=1}^{i} \frac{1}{4p_\ell} \leq \log t \cdot \ln t$ which is $O(\log^2 t)$, because for this value $t_{i+1} \leq t \cdot e^{-\ln t} = 1$. Plugging this upper bound on $\sum_{\ell=1}^{i} \frac{1}{4p_\ell}$ in the k_{i+1} lower bound we get that $k_{i+1} \geq k/2$. Therefore, k_j is always at least $k/2$ and so $t_j \geq k_j \geq k/2$. ■

3 Acknowledgments

The first author would like to thank Kamal Jain and Kunal Talwar for some initial discussions on the buy-at-bulk k-Steiner tree problem.

References

[1] M. Andrews, *Hardness of buy-at-bulk network design.*, in Proceedings of the 45th Symposium on Foundations of Computer Science (FOCS '04), 2004, pp. 115–124.

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

[3] B. Awerbuch and Y. Azar, *Buy-at-bulk network design*, in Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS '97), IEEE Computer Society, 1997, p. 542.

[4] B. Awerbuch, Y. Azar, A. Blum, and S. Vempala, *New approximation guarantees for minimum-weight k-trees and prize-collecting salesmen*, SIAM J. Comput., 28 (1999), pp. 254–262 (electronic).

[5] J. Bar-Ilan, G. Kortsarz, and D. Peleg, *Generalized submodular cover problems and applications*, Theoretical Computer Science, 250 (2001), pp. 179–200.

[6] Y. Bartal, *On approximating arbitrary metrices by tree metrics*, in Proceedings of the thirtieth annual ACM symposium on Theory of computing (STOC '98), New York, NY, USA, 1998, ACM Press, pp. 161–168.

[7] A. Blum, R. Ravi, and S. Vempala, *A constant-factor approximation algorithm for the k mst problem (extended abstract)*, in Proceedings of the twenty-eighth annual ACM symposium on Theory of computing (STOC '96), New York, NY, USA, 1996, ACM Press, pp. 442–448.

[8] M. Charikar and A. Karagiozova, *On non-uniform multicommodity buy-at-bulk network design*, in STOC '05: Proceedings of the thirty-seventh annual ACM symposium on Theory of computing, New York, NY, USA, 2005, ACM Press, pp. 176–182.

[9] C. Chekuri, S. Khanna, and J. Naor, *A deterministic algorithm for the cost-distance problem*, in Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms (SODA '01), Philadelphia, PA, USA, 2001, Society for Industrial and Applied Mathematics, pp. 232–233.

[10] F. A. Chudak, T. Roughgarden, and D. P. Williamson, *Approximate k-MSTs and k-Steiner trees via the primal-dual method and lagrangean relaxation*, in Proceedings of the 8th International Conference on Integer Programming and Combinatorial Optimization (IPCO '01), 2001, pp. 60–70.

[11] J. Chuzhoy, A. Gupta, J. S. Naor, and A. Sinha, *On the approximability of some network design problems*, in Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms (SODA '05), Philadelphia, PA, USA, 2005, Society for Industrial and Applied Mathematics, pp. 943–951.

[12] J. Fakcharoenphol, S. Rao, and K. Talwar, *A tight bound on approximating arbitrary metrics by tree metrics*, J. Comput. System Sci., 69 (2004), pp. 485–497.

[13] U. Feige, G. Kortsarz, and D. Peleg, *The dense k-subgraph problem*, Algorithmica, 29 (2001), pp. 410–421.

[14] N. Garg, *Saving an epsilon: a 2-approximation for the k-mst problem in graphs*, in Proceedings of the thirty-seventh annual ACM symposium on Theory of computing (STOC '05), New York, NY, USA, 2005, ACM Press, pp. 396–402.

[15] A. Goel and D. Estrin, *Simultaneous optimization for concave costs: single sink aggregation or single source buy-at-bulk*, in Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms (SODA '03), Philadelphia, PA, USA, 2003, Society for Industrial and Applied Mathematics, pp. 499–505.

[16] S. Guha, A. Meyerson, and K. Munagala, *Hierarchical placement and network design problems*, in Proceedings of the 41st Annual Symposium on Foundations of Computer Science (FOCS '00), Washington, DC, USA, 2000, IEEE Computer Society, p. 603.

[17] S. Guha, A. Meyerson, and K. Munagala, *A constant factor approximation for the single sink edge installation problems*, in Proceedings of the thirty-third annual ACM symposium on Theory of computing (STOC '01), New York, NY, USA, 2001, ACM Press, pp. 383–388.

[18] A. Gupta, A. Kumar, M. Pal, and T. Roughgarden, *Approximation via cost-sharing: a simple approximation algorithm for the multicommodity rent-or-buy problem*, in Proceedings of the 44rd Symposium on Foundations of Computer Science (FOCS '03), IEEE Computer Society, 2003, p. 606.

[19] A. Gupta, A. Kumar, and T. Roughgarden, *Simpler and better approximation algorithms for network design*, in Proceedings of the thirty-fifth ACM symposium on Theory of computing (STOC '03), ACM Press, 2003, pp. 365–372.

[20] M. Hajiaghayi and K. Jain, *The prize-collecting generalized Steiner tree problem via a new approach of primal-dual schema*, in Proceedings of the 17th annual ACM-SIAM symposium on Discrete algorithms (SODA '06), Philadelphia, PA, USA, 2006, Society for Industrial and Applied Mathematics. To appear.

[21] M. Hajiaghayi, G. Kortsarz, and M. R. Salavatipour, *Polylogarithmic approximation algorithm for non-uniform multicommodity buy-at-bulk*. Manuscript, 2005.

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

[23] S. Khuller, A. Moss, and J. S. Naor, *The budgeted maximum coverage problem*, Inf. Process. Lett., 70 (1999), pp. 39–45.

[24] A. Kumar, A. Gupta, and T. Roughgarden, *A constant-factor approximation algorithm for the multicommodity rent-or-buy problem*, in Proceedings of the 43rd Symposium on Foundations of Computer Science (FOCS '02), Washington, DC, USA, 2002, IEEE Computer Society, p. 333.

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

[26] A. Meyerson, K. Munagala, and S. Plotkin, *Cost-distance: two metric network design*, in Proceedings of the 41st Annual Symposium on Foundations of Computer Science (FOCS '00), IEEE Computer Society, 2000, p. 624.

[27] A. Moss and Y. Rabani, *Approximation algorithms for constrained for constrained node weighted steiner tree problems*, in STOC '01: Proceedings of the thirty-third annual ACM symposium on Theory of computing, New York, NY, USA, 2001, ACM Press, pp. 373–382.

[28] G. Robins and A. Zelikovsky, *Improved Steiner tree approximation in graphs*, in Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms (SODA '00), Philadelphia, PA, USA, 2000, Society for Industrial and Applied Mathematics, pp. 770–779.

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

[30] C. Swamy and A. Kumar, *Primal-dual algorithms for connected facility location problems*, Algorithmica, 40 (2004), pp. 245–269.