A Note on Subadditive Network Design Problem

M. Bateni[a,*,1], M. Hajiaghayi[b]

[a] *Computer Science Department, Princeton University, 35, Olden Street, Princeton, NJ 08540*
[b] *AT&T Labs – Research, 180, Park Avenue, Florham Park, NJ 07932*

Abstract

We study approximation algorithms for generalized network design where the cost of an edge depends on the *identities* of the demands using it (as a monotone subadditive function). Our main result is that even a very special case of this problem cannot be approximated to within a factor $2^{\log^{1-\varepsilon}|D|}$ if D is the set of demands.

Key words: approximation algorithm, network design, hardness of approximation, subadditive cost function

1. Introduction

We consider a multi-commodity setting in which the cost of an edge depends on the identities of the commodities using it, rather than solely on the total flow passing through it. The cost function should be monotone subadditive. A function $f : 2^X \mapsto \mathbb{R}$ is *subadditive* if for any $A, B \subseteq X$, we have $f(A) + f(B) \geq f(A \cup B)$. A function $f : 2^X \mapsto \mathbb{R}$ is said to be *monotone* if $f(A) \leq f(B)$ for any $A \subseteq B \subseteq X$. We are given a set of demand pairs in a graph and the goal is to buy some of the edges so that we can satisfy all demand pairs by connecting them together. In particular, each demand pair (s, t) should end up having a path connecting vertices s and t. Then, we say that the demand uses the edges on this path. The total cost incurred is the sum of the costs of the edges. The cost of each edge is a monotone subadditive function of the pairs using it. More formally, we consider the following problem in its most general form.

Definition 1 (Subadditive Steiner forest (SSF))**.** *Given an undirected graph $G(V, E)$, a set of demand pairs $D \subseteq V \times V$ to be connected, and a monotone subadditive cost function $c_e : D \mapsto \mathbb{R}^+$ for each edge $e \in E$, we are to find a set of paths, one P_d for each demand pair $d = (s, t) \in D$ so as to connect s to t, minimizing the total cost which is $\sum_e c_e(\{d : e \in P_d\})$.*

We first note that the Buy-at-bulk Steiner forest (BBSF) [3] problem is a special case of SSF. By studying this special case, Andrews [1] gives a hardness of approximation result of $\Omega(\log^{1/4} N)$ for the uniform version of SSF and a result of $\Omega(\log^{1/2} N)$ for the non-uniform version of SSF, where N is the number of vertices. We later observe that the uniform version of SSF can be approximated to within a factor $O(\log n)$ of the optimal, using the same technique as in Awerbuch and Azar [3]. In addition, we prove that the dependency of cost on the actual demand pairs, rather than just the amount of flow, makes the problem much more difficult. The hardness result holds true for a much simpler problem. Hence in what follows, we formulate this simpler problem, which is interesting in its own right.

Our model is a generalization of the models in Hayrapetyan, Swamy and Tardos [15] as well as Awerbuch and Azar [3]. It can also model many prize-collecting problems, such as [8, 14, 15]. To back up the latter claim, we briefly explain how our framework models that of [14]. In the prize-collecting Steiner forest problem (PCSF), we are given a graph $G(V, E)$, edge weights w_e for each edge $e \in E$, and a set of demand pairs $\mathcal{D}$ associated with penalty values π_d for each $d \in \mathcal{D}$. The goal is build a forest F that connects a subset of demands $\mathcal{D}' \subseteq \mathcal{D}$, minimizing the cost of F plus the total penalty of the demands in $\mathcal{D} \setminus \mathcal{D}'$. We reduce this problem into an instance of SSF as follows. Start with G where the cost associated with any edge $e \in E$ is simply w_e if any demand uses them. For any demand $d = (s, t) \in \mathcal{D}$,

*Corresponding author
Email addresses: mbateni@cs.princeton.edu (M. Bateni), hajiagha@research.att.com (M. Hajiaghayi)

[1] Supported by a Gordon Wu fellowship as well as by NSF ITR grants CCF-0205594 and CCF-0426582, NSF CAREER award CCF-0237113, MSPA-MCS award 0528414, and NSF expeditions award 0832797.

add an edge e_d connecting s and t. The cost of the edge is π_d if this particular demand uses it, and infinity if any other demand tries to pass through it. It is not difficult to verify that solving (or approximating) the produced SSF instance is equivalent to that of the original PCSF instance.

The dependence on the actual commodities rather than the total flow is important in practice, because some demands might require similar resources for which we may pay only once. In addition to the multiple commodities considered (with their obvious motivation), we consider different cost functions for different edges. Such a need arises since availability of resources at different locations of the network can affect the cost of building an edge. This, for instance, may be due to the fact that different telecommunication companies serve different portions of the network. In one major application, we only have a small number of different cost functions. In this case, there are a few companies involved. Each has its own cost function and owns some of the edges in the network. The cost per unit length of an edge depends solely on its provider, and is independent of its location in the network.

It is customary to study the single-sink variant of the network design problems. Not only does this simplify the treatment, but it can still capture several applications of the problem. Here, all the demands share a common vertex. Hence, we have a rooted problem.

Definition 2 (Subadditive rooted Steiner tree (SRST))**.** *Given an undirected graph $G(V, E)$, a set of terminals $T \subseteq V$ to be connected to a root vertex $r \in V$, and a monotone subadditive cost function $c_e : T \mapsto \mathbb{R}^+$ for each edge $e \in E$, our goal is to find a set of paths, one P_t for each terminal $t \in T$ connecting t to r, so as to minimize the total cost defined as $\sum_e c_e(\{t : e \in P_t\})$.*

This definition captures, among other problems, Steiner tree, Single-sink buy-at-bulk network design and Group-cost distance problems [3, 11]. In order to give the hardness of approximation result for SRST, we introduce another problem—a special case of SRST—and present the hardness for that problem instead. While in SRST the cost of each edge depends on the terminals using the edge in an unrestricted way (besides the subadditivity condition), in the following problem, we have for each edge a partition system on terminals, and the cost of the edge is its length multiplied by the number of sets in the system using the edge; i.e., several terminals in the same set can use an edge by paying the cost only once. The partition systems group together demands with similar resource requirements (common equipment). In other words, at each edge, it only matters how many different types of equipments one needs to construct the link. This can also be motivated by probable business contracts. The following problem is a generalization of the Group-cost distance problem [11].

Definition 3 (Group-cost rooted Steiner tree (GCRST))**.** *Given an undirected graph $G(V, E)$, a set of terminals $T \subseteq V$ to be connected to a root vertex $r \in V$, a nonnegative length w_e for each edge $e \in E$, and a partition $\mathcal{Q}_e = \{Q_{ei}\}$ of terminals for each edge $e \in E$, we are to find a set of paths, one P_t for each $t \in T$ connecting t to r, such that the total cost, $\sum_e w_e |\{Q_{ei} : \exists t, e \in P_t, t \in Q_{ei}\}|$, is minimized.*

A special case of SSF is the uniform case where all cost functions are multiples of the same function; the latter can be thought of as the cost-per-length function. The following theorem is proved in Section 3.

Theorem 4. *Uniform SSF admits an $O(\log n)$-approximation algorithm, even if the cost function is given by an oracle.*

It is interesting that the algorithm is independent of the actual oracle, provided that we know the length of the edges. In other words, we come up with a solution that is guaranteed to be good with respect to *any* oracle. It will be nice to get a lower bound tighter than Andrews' for our (possibly more difficult) problem, i.e., Uniform SSF. There is still a gap between $\Omega(\log^{1/4} n)$ and $O(\log n)$. The large size of the complete description of the instance, and our resorting to oracle access might help us achieve this, using a combination of inapproximability and communication complexity lower bounds. On the other hand, we prove that without the partition system simplification (or even with sufficiently many different partition systems), the problem is too hard to tackle. The proof of the hardness result is presented in Section 2.

Theorem 5. *GCRST cannot be approximated to within a factor $2^{\log^{1-\varepsilon}|T|}$ unless* NP $\subseteq$ DTIME$(n^{\text{poly}\log n})$*. This holds even if the number of different partition systems is only logarithmic in $|E|$.*

We emphasize here that it is crucial to have the dependence on the identities of commodities passing through an edge to get the strong hardness result. On the other hand, our positive result is more general than that of Awerbuch and Azar [3] in the same sense.

Other nice problems we hope to solve are some special cases of GCRST. The most important case is a Hierarchical GCRST, in which the available partitions have to be refinements of each other. That is, they have to form a hierarchy. The special case of two levels is equivalent to Group-cost distance. Though the case of three levels of hierarchy is already interesting, we hope to solve the case of a constant number of hierarchy levels, or even achieve some approximation guarantee linear in the number of levels in general.

A little more ambitious would be to consider the case with a few number of different (unrelated) partition systems. This is motivated by our observation that there are only a few major telecommunication companies and each of them has a unique cost function for all its edges (except for the multiplicative length factor). We do not know anything even for the case of two partition systems, but we guess an approximation guarantee exponential in the number of different partition systems should be possible. The hardness result of Theorem 5 says this might be almost tight.

2. Hardness

In this section, we prove a hardness result for GCRST. The reduction has ideas similar to the hardness result for Generalized stochastic Steiner tree problem of Gupta, Hajiaghayi and Kumar [11]. The reduction starts from the MinRep instance of Kortsarz [16]. Roughly speaking, we are to assign some labels to each vertex of a bipartite graph such that certain consistency conditions hold for each edge in the graph. The goal is to minimize the total number of labels used.

Definition 6 (MinRep)**.** *Given are a bipartite graph $G(V, W, E)$, a set of labels L, and a partial function $f_e : L \mapsto L$ for each edge e. We are to pick a subset of labels $L_u \subseteq L$ for each vertex $u \in V \cup W$, such that for each edge $e = (v, w) \in E$, we have $f_e(l) = l'$ for some $l \in L_v$ and some $l' \in L_w$. The goal is to minimize the total number of labels used, i.e., $\sum_{u \in V \cup W} |L_u|$.*

We need the projection property of MinRep: each label in the first partition, i.e., V, has at most one counterpart in the second partition, i.e., W. Otherwise, we could use the more well-known LabelCover problem [17] for which there is a slightly better hardness result known due to [5]. The following is stated in [16] but is easily derived from [17, 7, 2, 18].

Theorem 7 (Kortsarz [16])**.** *Unless* NP $\subseteq$ DTIME($n^{\text{poly}\log n}$)*, MinRep admits no approximation ratio better than $2^{\log^{1-\varepsilon} n}$ for any $\varepsilon > 0$, where n is the size of the input, i.e., $n = |V| + |W| + |E|$.*

Now we can prove the main component of our hardness result.

Theorem 8. *GCRST is as hard as MinRep.*

Proof. From a given MinRep instance as defined above $(V, W, E, L, \{f_e\}_{e \in E})$, we construct the following instance for GCRST $(V', E', T, r, c, \{Q_e\}_{e \in E'})$. We first present the formal definition of the reduction, which is followed by the intuition and the proof.

The vertex set V' consists of seven levels. The edges E' go only between the consecutive levels. We use the level number as part of the reference to the vertices in V'. The first level consists of a single root vertex $r = (1, r)$ to which all the terminals T are about to be connected. The following levels correspond to $V_2 = \{2\} \times V$, $V_3 = \{3\} \times V \times L$, $V_4 = \{4\} \times W \times L$, $V_5 = \{5\} \times W$, $V_6 = \{6\} \times E$ and terminals $T = \{7\} \times E \times [m]$, respectively. Here m is a parameter to be fixed later and $[m]$ is used to denote the set $\{1, 2, \ldots, m\}$.

We now define the edges in E'. There are edges of length zero from r to $(1, v) \in V_2$, from $(5, w) \in V_5$ to $(6, e = (u, v)) \in V_6$ and from $(6, e) \in V_6$ to $(7, e, i) \in T$. We also have edges of length one from $(2, v) \in V_2$ to $(3, v, l) \in V_3$ and from $(4, w, l) \in V_4$ to $(5, w) \in V_5$. Finally, we have edges of length $1/\sqrt{m}$ between $(3, v, l) \in V_3$ and $(4, w, l') \in V_4$ for each $(5, e = (v, w)) \in V_5$. Refer to Figure 1 for illustration.

The remaining bit from the construction is the partition systems Q_e for edges $e \in E'$. Except for the edges between V_3 and V_4, we use the trivial partition system $\{T\}$ that groups all the terminals together. This means that using each such edge has a fixed cost as defined above. However, the edges between V_3 and V_4 have more complex partition systems that we shortly introduce. This will serve as the core of the consistency check in our reduction. Take two vertices $(3, v, l) \in V_3$ and $(4, w, l') \in V_4$ that correspond to the edge $e = (v, w) \in E$. Let e' be the edge connecting

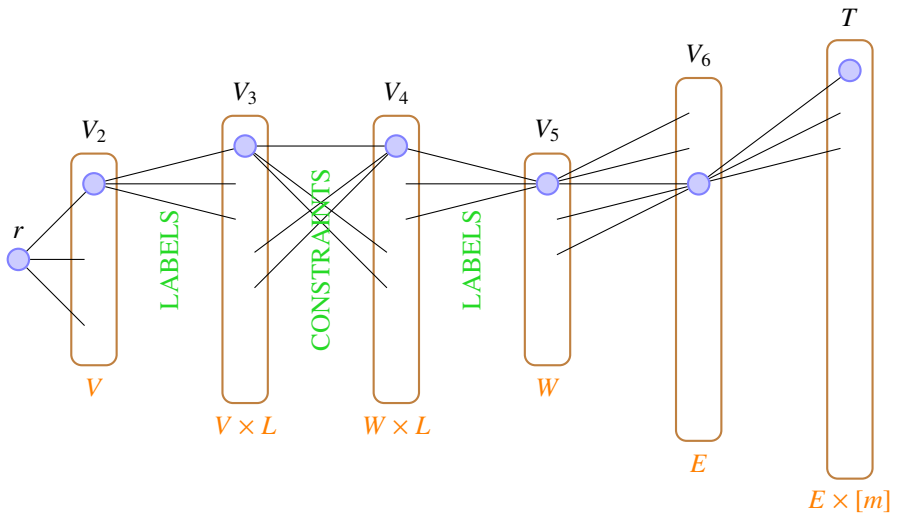

Figure 1: The reduction instance.

these two vertices in E'. Then $\mathcal{Q}_{e'}$ puts all the terminals in their own singleton sets except possibly for the terminals $T_e = \{7\} \times \{e\} \times [m]$. The terminals T_e are in the same set of $\mathcal{Q}_{e'}$ if $f_e(l) = l'$, otherwise, each has its own singleton set.

The intuition is that the edges of cost one (i.e., those between V_2 and V_3 or V_4 and V_5) are going to measure the cost of the MinRep instance (i.e., that the total number of labels used) and the customized partition systems for edges between V_3 and V_4 account for the consistency of the constraints. We pick m sufficiently large so that any good solution is forced to satisfy the consistency checks. Roughly speaking, if the m copies of an edge $e \in E$ do not use a *compatible* edge between V_3 and V_4 (to be compatible, this edge has to correspond to the same pair of vertices and the labeling should be valid), they will incur a huge cost. Because, by using a compatible edge, they only pay once, whereas they will need to pay m times if they use other edges. In the following, we show how the solution to the MinRep instance is essentially equivalent to that of the produced GCRST instance.

Suppose we have a solution $\{L_u\}_{u \in V \cup W}$ to MinRep of cost C. Each constraint $e = (v, w) \in E$ is satisfied by at least a pair of good labels (l, l'), i.e., $f_e(l) = l'$. Then a terminal $(7, e, i) \in T$ can use the following path: it first goes to $(6, e)$, then to $(5, w)$, $(4, w, l')$, $(3, v, l)$, $(2, v)$ and eventually to r. The new cost would be $C + |E|/\sqrt{m}$, since we pay one unit for any label used and $1/\sqrt{m}$ to allow terminals of the form $(7, e, i)$ for each $e \in E$ to pass through a compatible edge between V_3 and V_4. As we will see, m will be chosen so that the additive term above is negligible.

In any solution to GCRST, consider the paths terminals $(7, e, i)$ use for some $e \in E$. If one of them uses a good labeling of the constraint corresponding to e, then all the others can also use the same path without increasing the cost; this is due to the construction of $\mathcal{Q}$. So consider this is the case, i.e., either none or all of them use a good labeling. The latter case is what we desire, but in the former case, each of these terminals is using an edge connecting V_3 and V_4. These edges will charge each such terminal separately with a total cost of $m \cdot \frac{1}{\sqrt{m}} = \sqrt{m}$. Notice that a feasible MinRep instance always has a solution of size at most $2|E|$. Thus by choosing $m = \Theta(|E|^4)$, we ensure that not using the proper labeling for these terminals would make us incur substantial cost.

Now, consider a solution to GCRST which has the above-mentioned property that middle edges used are suitable labels for the terminals using them. Then, each terminal has a path to the root, which at some point for the first time goes from some $(4, w, l')$ to a $(3, v, l)$. The next step cannot be any vertex other than $(2, v)$ (because f_e is a partial function for $e = (v, w)$). Hence, both $(4, w, l')$ and $(3, v, l)$ are accounted for in the solution; i.e., they have their edges of cost one. Therefore, a solution of value C' for GCRST can be turned into a solution of value at most $C' - |E|/\sqrt{m}$ for MinRep. Ignoring the negligible effect of the additive term, the latter has about the same cost.

Notice that $|T| = m|E|$, $|V'| = O((|V| + |W|) \cdot |L| + |T|)$, and $|E'| = O(|V'|^2)$. The size of the instance grows polynomially and the reduction can be performed in polynomial time. □

Combined with Theorem 7, this establishes a hardness of factor $2^{\log^{1-\varepsilon}|T|}$ for GCSRT. Although we didn't try to

optimize m, the above construction could be improved so that $m = O(|V| \cdot |E|) = O(|E|^2) = O(n^2)$, where n is the total number of vertices in the MinRep instance. None of these improvements, though, has much effect in the final hardness. The number of different partition systems used in our construction is $\Theta(|E|)$. Next, we show how one can reduce this considerably in order to conclude that the hardness is in a sense stemming from the number of terminals rather than the different partition systems.

Proof of Theorem 5. Everything is the same as in proof of Theorem 8, barring the single issue of building the *set-dependent* edges between V_3 and V_4. Recall these were the edges that actually enforced the labeling constraints. There were $|E| + 1$ different partition systems used. One was the trivial partition system grouping every vertex together. Any other partition system corresponds to an edge $e \in E$; let is be called $\mathcal{Q}^e$. Consider an edge e' of length one having a partition system $\mathcal{Q}^e$. All terminals of the form $(7, e, i)$ are placed in the same set of $\mathcal{Q}^e$ whereas other vertices have their own sets. Ideally, they were meant to allow terminals of the above form pass with a small cost while disallowing any other terminals. We try to model these E different partition systems using only $O(\log |E|)$ partition systems.

Let original edges be numbered from zero to $|E| - 1$. This requires numbers of length $O(\log |E|)$. We provision two partition systems for each fixed bit position k in the representation: $\mathcal{Q}^{k,1}$ that groups together the items which have a 1 in this position, and $\mathcal{Q}^{k,0}$ that groups the zeros together. Any edge using a partition system $\mathcal{Q}^e$ is replaced by a path of length $K = O(\log |E|)$. Each such edge has length $1/K$. The k-th edge on the path has partition system $\mathcal{Q}^{k,x_k}$ where x_k is the value of the k-th bit in the binary representation of e. Now all the terminals of the form $(7, e, i)$ can use this path and pay a total cost of one unit. However, we claim if any other group of m' terminals uses this edge, they have to pay at least m'/K^2 units. We will prove this shortly, but let us first see why this suffices to finish the proof. The same argument as the previous proof works with the following changes. Should none of the terminals of the form $(7, e, i)$ for a fixed $e \in E$ use a compatible edge between V_3 and V_4, the cost of the solution will be at least $\sqrt{m}/O(\log^2 |E|)$. Compared to the previous proof, we lost a polylogarithmic factor which is not important due to the choice of m. We would argue that such a solution is much costlier than a trivial solution of cost $2|E|$. Hence, we do not consider these solutions.

We now prove the claim: a group of m' terminals T' of the form $(7, e, i) \in T$ where $e \neq e'$ that pass through the replacement of $\mathcal{Q}^{e'}$ has to pay at least m'/K^2. Recall that the replacement consists of K edges. Thus, we prove that these terminals use more than m'/K groups at one of these edges. Suppose for the sake of contradiction that at most m'/K groups are used at each edge. We prove via induction that T' has at least $m' - k(m'/K - 1)$ terminals $(7, e, i)$ where the representation of e and e' share the same bits at the first k positions. Setting $k = K$ guarantees us $K > 0$ terminals in T' corresponding to the edge e' which is a contradiction, thus disproves the supposition. Let us prove the inductive claim here. The base case is clear for $k = 0$. Suppose the claim is true for some $k < K$. Notice that in the $(k + 1)$-th edge, T' uses no more than m'/K groups. Hence, at most $m'/K - 1$ out of these $m' - k(m'/K - 1)$ terminals can have a different $(k + 1)$-th bit from e', which finishes the inductive argument. □

3. Algorithm

Uniform SSF is a generalization of Uniform buy-at-bulk network design. Unlike the simpler cases solved in [3, 9, 13, 12], the optimal solution for this case need not be a tree. In fact, for the simplest case of Group-cost distance (even without the global partition), one can come up with a counterexample, say, as in Figure 3. Here, the top node is the root and the nodes at the bottom-most row are the terminals whose partitioning is specified by their tokens' count; i.e., the two middle terminals belong to the same partition set whereas the others have their own singleton sets. The optimal tree solution has cost 8; the total cost does not change whether we use the horizontal edge or not. However, the optimal non-tree solution has cost 7: one of the middle terminals uses the horizontal edge to merge with its pair so that the pair can pay the cost of the last edge only once. The ratio between the best tree and non-tree solutions can be easily improved to $4/3 - \varepsilon$ by having more pairs of terminals in this example. More precisely, we have k pairs of terminals (a_i, b_i). The root r is connected to nodes c_i. Each vertex c_i is then connected to its respective a_i and b_i. In addition, the nodes c_i form a complete graph. The edges incident on the root have cost two, those incident on terminals have zero cost, and the other edges cost one unit each. The partition system groups all a_i nodes in one set and every other node in a singleton. It is easy to verify that the cost of the optimal tree solution is $4k$ while the optimal non-tree solution has cost $2(k + 1) + (k - 1) = 3k + 1$. We do not know how wide this *treeness gap* can be.

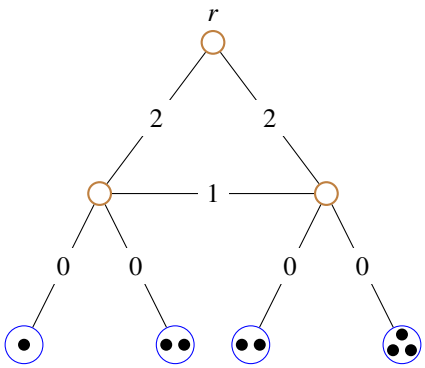

Figure 2: Example for non-tree optimal solution.

While we seek a constant factor approximation in the single-sink case, it is worthwhile to note that the above example poses a major barrier. The only LP with a constant gap for the problem assumes the tree optimality property for writing the LP. We have to prove an upper bound on the treeness gap to use this method. The other approaches giving constant factor guarantees heavily rely on the structure of the edge costs, say as a (relatively) small number of different edge types. For instance, having oracle access to a submodular function, though, barely gives us any information about the structure of the function.

Nonetheless, using same ideas of [3], we can get a simple $O(\log n)$-approximation algorithm. For the sake of completeness, we present the details of the algorithm in this paper, especially since our algorithm is more general in the sense that the cost of an edge depends on the identities of the demands using it (this is not the case for them). We first use FRT's method of probabilistic embedding into trees to build a random tree taking edge lengths w_e as distances. This method was first introduced by [4] but the tight guarantee was proved in [6]. A graph G is approximated by a random tree T with the following two properties: the distances may only increase in the tree, i.e.,

$$d_G(u, v) \leq d_T(u, v) \quad \forall u, v \in V, \tag{1}$$

where $d_G(u, v)$ for a graph G denotes the length of the shortest path between u and v in G. Furthermore, the growth in distances is within certain bounds. In particular,

$$\mathbb{E}[d_T(u, v)] \leq O(\log n) \cdot d_G(u, v) \quad \forall u, v \in V, \tag{2}$$

where the expectation is over the choice of the tree T. Next we remove vertices that do not correspond to vertices of G, using the ideas in Gupta [10]. Afterwards, we just find the straightforward solution in the tree case: take the unique path P'_d connecting the endpoints of each demand $d \in D$. Finally, we translate this solution to a solution for the original graph G. To do this, we replace each edge of the path P'_d with the shortest path connecting its endpoints in G. This produces a walk which we then simplify to get the path P_d. Let $\mathcal{P}_e$ for an edge $e = (u, v)$ of T be the shortest path in G between u and v. For a path p in T, we define $\mathcal{P}(p)$ as the walk formed by replacing each edge e of p by $\mathcal{P}_e$. Then $\mathcal{P}^s(p)$ is the result of eliminating cycles of $\mathcal{P}(p)$.

We shortly show that the cost of this solution is at most a factor $O(\log n)$ times the optimal. In fact, this also works if we have *subadditive* functions and in the *multicommodity* setting, which is more general than the previous results as our cost functions distinguish the terminals.

Proof of Theorem 4. We show that the algorithm in Figure 3 achieves the goal. To this end, we first show the cost of the solution in the probabilistic tree T is not too large.

Take the optimal solution in the original graph G. An edge $e \in E_G$ is used by a subset F_e of demand pairs. Then the cost of the optimal solution is

$$\mathsf{OPT} = \sum_{e \in E_G} w_e \cdot g(F_e),$$

Algorithm UnfiromSSF-Solver
Input: *Graph $G(V, E_G)$, demands D, edge lengths $\{w_e\}_{e\in E}$ and cost per length function $g(.)$*
Output: *Set of paths $\{P_d\}_{d\in D}$*

1. Approximate the weighted graph $G(V, E_G, \{w_e\}_{e\in E_G})$ by a random tree T' generated via the method of [6].
2. Remove the Steiner nodes of T' to get $T = (V, E_T)$ using the procedure of [10]
3. For any demand $d = (s, t) \in D$, pick the unique path P'_d connecting s and t in T.
4. Produce the final solution as $P_d = \mathcal{P}^s(P'_d)$ for each $d \in D$.

Figure 3: The algorithm for UniformSSF.

where $g(.)$ is the oracle function common to all the edges and w_e is the particular length of an edge e. Now in T, let each edge e' serve the terminals

$$\tilde{F}(e') = \bigcup_{p_e \ni e'} F_e,$$

where p_e is the unique path in T between endpoints of the edge e (of the original graph). It is straightforward to see such a provisioning will satisfy all the demand pairs. The expected cost of this solution is

$$\begin{aligned}\sum_{e'\in E_T} w(e') \cdot g(\tilde{F}(e')) &= \sum_{e'\in E_T} w(e') \cdot g\left(\bigcup_{p_e\ni e'} F(e)\right) \overset{(3)}{\leq} \sum_{e'\in E_T} w(e') \cdot \sum_{p_e\ni e'} g(F(e)) \\ &= \sum_{e\in E_G} g(F(e)) \cdot \sum_{e'\in p_e} w(e') \overset{(4)}{=} \sum_{e\in E_G} g(F(e)) O(\log n) w(e) = O(\log n)\mathsf{OPT},\end{aligned}$$

where (3) follows from subadditivity of $g(.)$ and (4) is the second guarantee of the probabilistic embedding (2).

The solution we find on the tree is the optimal solution, since any solution is forced to make each demand pair use all the edges in the unique path connecting them. It only suffices to prove translating the solution of T to that of G does not increase the cost. Each P'_d is translated to $\mathcal{P}^s(P'_d)$. The optimal solution for T costs

$$\begin{aligned}\sum_{e'\in E_T} w(e') \cdot g\left(\bigcup_{\substack{d\in D\\ e'\in P'_d}} \{d\}\right) &\overset{(5)}{\geq} \sum_{e'\in E_T} \sum_{e\in\mathcal{P}(e')} w(e) \cdot g\left(\bigcup_{\substack{d\in D\\ e'\in P'_d}} \{d\}\right) = \sum_{e\in E_G} w(e) \cdot \sum_{\substack{e'\in E_T\\ e\in\mathcal{P}(e')}} g\left(\bigcup_{\substack{d\in D\\ e'\in P'_d}} \{d\}\right) \\ &\overset{(6)}{\geq} \sum_{e\in E_G} w(e) \cdot g\left(\bigcup_{\substack{e'\in E_T\\ e\in\mathcal{P}(e')}} \left[\bigcup_{\substack{d\in D\\ e'\in P'_d}} \{d\}\right]\right) \overset{(7)}{=} \sum_{e\in E_G} w(e) \cdot g\left(\bigcup_{\substack{d\in D\\ e\in\mathcal{P}(P'_d)}} \{d\}\right) \\ &\overset{(8)}{\geq} \sum_{e\in E_G} w(e) \cdot g\left(\bigcup_{\substack{d\in D\\ e\in\mathcal{P}^s(P'_d)}} \{d\}\right) = \sum_{e\in E_G} w(e) \cdot g\left(\bigcup_{\substack{d\in D\\ e\in P_d}} \{d\}\right),\end{aligned}$$

which is the value of the solution we produce. In the above derivation, (5) is a result of (1); subadditivity and monotonicity of $g(.)$ give (6) and (8) respectively. Finally, Equation (7) is a restatement of definition of $\mathcal{P}(P'_d)$. □

References

[1] Matthew Andrews. Hardness of buy-at-bulk network design. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 115–124, 2004.

[2] Sanjeev Arora, László Babai, Jacques Stern, and Z. Sweedyk. The hardness of approximate optimia in lattices, codes, and systems of linear equations. In *34th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 724–733, 1993.

[3] Baruch Awerbuch and Yossi Azar. Buy-at-bulk network design. In *Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 542–547, 1997.

[4] Yair Bartal. Probabilistic approximations of metric spaces and its algorithmic applications. In *Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 184–193, 1996.

[5] Irit Dinur and Shmuel Safra. On the hardness of approximating label-cover. *Information Processing Letters*, 89(5):247–254, 2004.

[6] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *Journal of Computer and System Sciences*, 69(3):485–497, 2004.

[7] Uriel Feige and László Lovász. Two-prover one-round proof systems: Their power and their problems (extended abstract). In *Proceedings of the Twenty Fourth Annual ACM Symposium on Theory of Computing (STOC)*, pages 733–744, 1992.

[8] Michel X. Goemans and David P. Williamson. A general approximation technique for constrained forest problems. In *Proceedings of the Third Annual ACM/SIGACT-SIAM Symposium on Discrete Algorithms (SODA)*, pages 307–316, 1992.

[9] Sudipto Guha, Adam Meyerson, and Kamesh Munagala. A constant factor approximation for the single sink edge installation problems. In *Proceedings on 33rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 383–388, 2001.

[10] Anupam Gupta. Steiner points in tree metrics don't (really) help. In *Proceedings of the Twelfth Annual Symposium on Discrete Algorithms (SODA)*, pages 220–227, 2001.

[11] Anupam Gupta, MohammadTaghi Hajiaghayi, and Amit Kumar. Stochastic steiner tree with non-uniform inflation. In *Proceedings of the 9th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, pages 134–148, 2007.

[12] Anupam Gupta, Amit Kumar, Martin Pál, and Tim Roughgarden. Approximation via cost-sharing: A simple approximation algorithm for the multicommodity rent-or-buy problem. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 606–615, 2003.

[13] Anupam Gupta, Amit Kumar, and Tim Roughgarden. Simpler and better approximation algorithms for network design. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing (STOC)*, pages 365–372, 2003.

[14] MohammadTaghi Hajiaghayi and Kamal Jain. The prize-collecting generalized steiner tree problem via a new approach of primal-dual schema. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm (SODA)*, pages 631–640, 2006.

[15] Ara Hayrapetyan, Chaitanya Swamy, and Éva Tardos. Network design for information networks. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms (SODA)*, pages 933–942, 2005.

[16] Guy Kortsarz. On the hardness of approximating spanners. *Algorithmica*, 30(3):432–450, 2001.

[17] Carsten Lund and Mihalis Yannakakis. On the hardness of approximating minimization problems. *Journal of the ACM*, 41(5):960–981, 1994.

[18] Ran Raz. A parallel repetition theorem. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing (STOC)*, pages 447–456, 1995.