Network Design with Coverage Costs*

Siddharth Barman[1], Shuchi Chawla[2], and Seeun Umboh[3]

1 **California Institute of Technology, U.S.**
`barman@caltech.edu`
2 **University of Wisconsin – Madison, U.S.**
`shuchi@cs.wisc.edu`
3 **University of Wisconsin – Madison, U.S.**
`seeun@cs.wisc.edu`

Abstract

We study network design with a cost structure motivated by redundancy in data traffic. We are given a graph, g groups of terminals, and a universe of data packets. Each group of terminals desires a subset of the packets from its respective source. The cost of routing traffic on any edge in the network is proportional to the total size of the distinct packets that the edge carries. Our goal is to find a minimum cost routing. We focus on two settings. In the first, the collection of packet sets desired by source-sink pairs is laminar. For this setting, we present a primal-dual based 2-approximation, improving upon a logarithmic approximation due to Barman and Chawla (2012) [7]. In the second setting, packet sets can have non-trivial intersection. We focus on the case where each packet is desired by either a single terminal group or by all of the groups. This setting does not admit an $O(\log^{\frac{1}{4}-\gamma} g)$-approximation for any constant γ under a standard assumption; we present an $O(\log g)$-approximation when the graph is unweighted.

Our approximation for the second setting is based on a novel spanner-type construction in unweighted graphs that, given a collection of g vertex subsets, finds a subgraph of cost only a constant factor more than the minimum spanning tree of the graph, such that every subset in the collection has a Steiner tree in the subgraph of cost at most $O(\log g)$ that of its minimum Steiner tree in the original graph. We call such a subgraph a group spanner.

1998 ACM Subject Classification F.2.0 Analysis of Algorithms and Problem Complexity

Keywords and phrases Network Design, Spanner, Primal Dual Method, Traffic Redundancy

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2014.48

1 Introduction

Some of the classical applications of the theory of algorithms are in transportation and commodity networks: how should commodities be transported from where they are manufactured to where they are consumed? How should pipelines be laid to be most effective at balancing costs with requirements? These questions have spawned many basic problems and theorems in the area of approximation algorithms: network flow, traveling salesman, Steiner tree, flow-cut gaps, etc. Over time, solutions to these problems have come to be applied to a different class of networks, namely communication networks. At a basic level, the problems in communication networks are similar: how should data be routed from its sources to its destinations? How should networks be designed to be able to handle different kinds of workload and traffic patterns? However, the underlying commodity in these

* This work was partially supported by NSF grants CNS-0846025 and CCF-1101470. Siddharth Barman gratefully acknowledges the support of the Linde/SISL postdoctoral fellowship.

17th Int'l Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX'14) /
18th Int'l Workshop on Randomization and Computation (RANDOM'14).
Editors: Klaus Jansen, José Rolim, Nikhil Devanur, and Cristopher Moore; pp. 48–63
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

networks—data—is fundamentally different from physical commodities. Unlike the latter, data can be compressed, encoded, or replicated, at virtually no cost. Network algorithms that do not exploit these properties fail to utilize the entire capacity of the network.

The last few years have seen a rapid growth in "content aware" network optimization solutions, both within the academic literature (see, e.g., [1, 26], and references therein) as well as in the form of commercial technologies [9, 24]. One of the functionalities that these technologies provide is to remove duplicate traffic from the network. Every router in the network equipped with such a technology keeps track of recently seen traffic. When duplicates are detected, a single copy of the duplicated data is sent forward along with a short message containing instructions for replication at the next router. This defines a cost function on every link in the network, where the cost of carrying data is proportional to the number (or total size) of *distinct* packets that the link carries; in other words, it is a *coverage function* over the set of traffic streams that use the link. We study network design problems within this context.

We consider the following framework. We are given a weighted network, and multiple *commodities*, each with a source and several possible destinations that we collectively call terminals. Each commodity is composed of a number of different data packets drawn from a universe of packets; we call these sets of packets *demands*. Importantly, there is redundancy in traffic—different commodities may overlap in the sets of packets they contain, and so can benefit from using common routes. Our goal is to find a minimum cost routing for the given traffic matrix, assuming that we can buy bandwidth at a fixed rate on every edge. Formally, our solution specifies for each commodity a routing tree spanning all of the terminals for this commodity. The cost of this solution on any particular edge is proportional to the total size of the distinct packets that the edge carries. This problem was introduced in [7] where it was called redundancy aware network design.

Network design with coverage costs displays the same short-routes-versus-shared-routes tradeoff present in several classical network design problems with nonlinear costs, such as rent-or-buy network design [19, 15], access network design [4], and buy-at-bulk network design [5, 16, 21, 28]. However there are fundamental differences. The buy-at-bulk cost model is inspired by economies of scale in a physical commodity network—the volume of traffic that an edge carries is the sum of the volumes that the different commodities impose on it and the routing cost on the edge is a concave function of the total volume of traffic. On the other hand, in our setting, the volume of traffic itself is lowered due to the inherent nature of data traffic. In particular, this means that the savings achieved depend on the contents of the traffic and not just its quantity. We not only need to bundle traffic streams as much as we can, but we also need to decide the right sets of traffic streams to bundle. Consequently, the approximability of the problem also depends on the extent and manner in which different commodities share packets. When every source-sink pair in the network demands a distinct packet, that is, there is no data redundancy in the network, the problem reduces to finding the shortest route for each pair. When all of the demands are identical, the problem reduces to finding a single optimal Steiner forest over all of the terminal sets.

Laminar and sunflower demands. In this paper we focus on two special cases of the network design problem with coverage costs—the *laminar demands* setting, and the *sunflower demands* setting. In the laminar demands setting the packet sets corresponding to the commodities form a laminar family: the packet sets of any two commodities are either completely disjoint or one contains the other. There is a natural hierarchy over commodities in this setting and any commodity can use for free an edge that is being used for another commodity that "dominates" it. So we may favor long routes for a commodity if those routes share edges with a dominating commodity, in comparison to shorter ones that do not share edges. Less intuitively, it may be useful to pick similar routes for two commodities with disjoint packets sets if a portion of the shared route can be used for a commodity that dominates both. Consequently, commodities that are higher up in the hierarchy are in some sense more important than commodities that are lower in the hierarchy.

Non-laminar settings, where packet sets can have arbitrary intersection, also display sharing of paths among similar as well as dissimilar commodities. However, we cannot exploit any natural ordering over commodities in determining which paths to use so we require techniques that are very different from those used for the laminar demands setting. As a first step, we study the simplest setting that captures the complexity introduced by non-trivial intersections. In the *sunflower demands* setting, every collection of demands has the same intersection. In other words, there is a common set of packets that belongs to every commodity, and every other packet belongs to exactly one commodity. A simple example of this setting is where each demand is of the form $\{0, i\}$; here 0 denotes the common packet, and i denotes the packet belonging only to commodity i. Once again our goal is to construct a routing tree for each commodity of minimum total cost. The cost of the collection of routing trees has two components. The first corresponds to the total size of the union of the routing trees: we pay for the cost of routing the common packets on this entire subgraph. The second corresponds to the costs of the individual trees, weighted by the sizes of their respective unique packets. A natural interpretation for the cost structure is as follows: for each edge, there is a fixed cost (per unit length) for buying the edge before it can be used for routing, and a variable cost (per unit length) that depends on the number of commodities being routed on it.

1.1 Our results and techniques

We present a primal-dual based 2-approximation for the laminar demands setting, improving upon a logarithmic approximation by [7] and matching the approximation factor for the Steiner forest problem, which is a special case. The sunflower demands setting, on the other hand, is much harder. In particular, it captures as a special case the buy-at-bulk network design problem with a single cable type with linear cost; this special case was shown in [2] to be inapproximable to within a factor polylogarithmic in g under standard assumptions, where g is the number of commodities (see Theorem 8 in Section 4). We present an $O(\log g)$ approximation for this problem under two further assumptions[1]: (1) the graph is unweighted; (2) every node is a terminal. We leave open the question of designing an approximation for the general sunflower demands setting. Note, however, that an $O(\log n)$ approximation can be obtained by first embedding the network into a tree with low distortion and then solving the problem on the tree; Here n is the number of nodes in the network. Our $O(\log g)$ approximation is based on a novel spanner-type construction described below that may be of independent interest. We now describe our techniques in more detail.

[1] We note that the first assumption by itself, i.e. the graph is unweighted, is without loss of generality: since our approximation is with respect to the total cost of the solution, and not with respect to the number of edges in it, we can break up each long edge into edges of equal size by introducing new nodes. However, the additional assumption that every vertex belongs to some terminal set disallows this sort of transformation.

Sunflower demands. A standard approach in network optimization is to approximate a given network by a subgraph that is much cheaper or sparser than the entire graph, and yet faithfully captures some essential property of the graph. For example, spanners [23] are low-cost subgraphs that approximately capture shortest path distances between every pair of points in the graph. Likewise, cut- and flow-sparsifiers [22, 20] are sparse subgraphs that approximate cuts and flows in the graph respectively. Network design with coverage costs defines another such graph sparsification problem that may be of independent interest. In particular, for a given solution to the network design problem, consider partitioning the edges into sets that carry a particular packet. Each such set is a Steiner forest over the terminal sets that demand that packet. Our goal is to find a solution that minimizes a weighted sum of the sizes of these Steiner forests. One way of doing so may be to find a subgraph that induces Steiner forests over each respective set of terminals corresponding to a single packet, that are simultaneously approximately minimal for their corresponding instances. This approach is particularly relevant for the sunflower demands setting. In that setting, the Steiner forest corresponding to the common packets is the entire subgraph itself, whereas the forest corresponding to packets unique to a commodity is simply the routing tree constructed for that commodity. We therefore ask: is there a subgraph that $\alpha-$approximates the size of the minimum Steiner forest over the union of all terminal sets, and at the same time induces a Steiner tree over each individual terminal set that is within a factor of β of the smallest such tree? We call such a subgraph an (α, β) *group spanner*. Group spanners generalize spanners: if for every pair of nodes in the graph our instance contains a terminal set comprising of the two nodes, then a group spanner for the instance simultaneously approximates the shortest path distances between every pair of nodes. The factor β is called the stretch of the spanner.

The main technical component in our approach for the sunflower demands setting is a construction for group spanners in unweighted graphs where the union of all terminal sets spans the entire graph. Our construction achieves an $(O(1), O(\log g))$ approximation. This implies an $O(\log g)$ approximation for the sunflower demands setting under those assumptions. A widely-believed conjecture of Erdős [11] and others (see [29] for a longer discussion) implies that no (α, β) group spanners with $\alpha\beta = o(\log g)$ exist, so our construction achieves the optimal tradeoff between size and stretch. The problem of extending our construction to arbitrary weighted graphs remains an interesting open question.

Laminar demands. To form intuition for this setting consider an instance with k different packets and $k + 1$ commodities: for $i \leq k$ the demand set of commodity i contains only packet i, and demand set of commodity $k + 1$ contains all of the k packets. Suppose also that every commodity has a single source and a single sink. Then, one approach to solving the problem is to first find a least cost path for commodity $k + 1$, and then find least cost paths for the remaining commodities using the edges in the first path for free. This approach misses solutions where a slightly longer path for commodity $k + 1$ is much more cost efficient for the remaining commodities than the shortest path for $k + 1$. An alternative is to first find shortest paths for commodities 1 through k, and then find the least cost path for commodity $k + 1$ that can use edges in previously picked paths at a cheaper cost. This misses solutions where picking slightly longer paths for commodities 1 through k leads to a greater sharing of the edges. The first approach is indeed the approach analyzed in [7] for the special case of the problem where there is a single source that belongs to all of the terminal sets. That paper shows that in any single source laminar demands setting routing commodities in order of decreasing sizes of demand sets achieves an $O(\log k)$ approximation where k is the number of different packets in the universe[2].

We extend and improve the result of [7] to obtain a 2-approximation for the laminar demands setting with arbitrary terminal sets. Our approach is a hybrid of the two described above and is based on a non-standard LP formulation of the problem. Our LP encodes for each demand set the edges that carry this demand set but no other demand set that dominates it. We then apply a primal-dual approach. At a high level, we first consider commodities in *increasing* order of the sizes of their demand sets. However, instead of committing to a single path for each commodity before considering the next, we keep around a collection of all possible near-optimal paths for the smaller demand sets before considering choices for the larger demand sets. Then in a second pass, we finalize a single path (tree) for each commodity, considering commodities in *decreasing* order of sizes of their demand sets. That is, we commit to paths for the larger demand sets before finalizing paths for the smaller demand sets. The duals constructed for each commodity give a succinct description of all possible short paths connecting the source and the sink for that commodity. After having constructed all of the duals, we perform a reverse delete step that finalizes paths for commodities starting from the one with the largest demand and moving on to smaller demand sets.

1.2 Connections to other network optimization problems

The cost structure in the network design problem we consider is uniform in the sense that costs on different edges are related through constant factors. Obtaining a randomized $O(\log n)$ approximation for network design problems with a uniform cost structure is often easy: we can use the tree embeddings of Bartal [8] and Fakcharoenphol et al. [12] to convert the graph into a distribution over trees such that distances between nodes are preserved to within logarithmic factors in expectation. Then the expected cost of the optimal routing over the (random) tree is related within logarithmic factors to the cost of the optimal routing over the graph. Moreover, the problem is easy to solve on trees, because there is a unique path between every pair of nodes. We achieve much better approximation factors. For the laminar demands setting, we obtain a 2-approximation. For the sunflower demands setting, our approximation factor is $O(\log g)$; note that g is always at most n, and in most applications should be much smaller.

As mentioned earlier, network design with coverage costs is closely related but incomparable to other models of network design with uniform costs that display economies of scale. This includes, e.g., the uniform buy-at-bulk [5, 16, 21, 28], rent-or-buy [19, 15], and access network design [4, 14] problems. For all of these problems constant factor approximations are known in the uniform costs setting for the special case where all of the commodities share a common source. In the multi-commodity setting, i.e., with distinct sources and sinks, the rent-or-buy network design problem admits a 2-approximation [19, 15], but the buy-at-bulk network design problem is hard to approximate within poly-logarithmic factors [3].

Cost models specific to communication networks have been considered before in network design. Hayrapetyan et al. [17] study a single-source network design problem in which the cost on an edge is a monotone submodular function of the commodities that use the edge. They obtain an $O(\log n)$ approximation via tree embeddings [8, 12], where n is the number of vertices in the graph. The cost structure that we consider is a special case of the one

[2] In fact, after a slight transformation of the instance, the same approximation can be obtained by approximating the minimum Steiner tree for every demand set separately and combining the solutions

in [17] (coverage functions are submodular). However, unlike [17] we assume that terminals sets are arbitrary (in particular, they do not share a common source). Moreover, we obtain stronger approximation guarantees.

Shmoys et al. [25] study a facility location problem with a cost structure very similar to that in our sunflower demands setting. In their model, the cost of opening a facility has two components: a fixed cost (similar to the cost of routing the common packets in our setting), and a service specific cost (similar to the cost of routing other packets in our setting). They present a constant factor approximation for facility location with this cost structure. Svitkina and Tardos [27] further extend this to a facility location problem with hierarchical costs, again presenting a constant factor approximation. Extending our results to more general non-laminar coverage functions including hierarchical costs is an interesting open problem.

As mentioned earlier, a main component in our approach for the sunflower demands setting is a construction for group spanners in unweighted graphs. Group spanners generalize graph spanners. Low-stretch spanners have a number of applications, including distributed routing using small routing tables and in computing near-shortest paths in distributed networks (see [23] and references therein). In unweighted graphs it is well known that the size of the smallest spanner with multiplicative stretch k is equal to the maximum number of edges in a graph with girth at least $k+1$; this is known to be $O(n^{1+O(1/k)})$, and is conjectured tight. Our result is consistent with this bound: when the number of commodities g is equal to the number of vertex pairs, we get an $O(\log g) = O(\log n)$ stretch with a spanner of size $O(n)$. Other work on spanners has focused on additive stretch and weighted graphs (see, e.g., [10, 23, 30]).

Group spanners also generalize shallow-light spanning trees. The latter is a subgraph that is simultaneously an approximately-minimum spanning tree of the given graph, as well as an approximate-shortest-paths tree with respect to a given source node. Consider an instance with a special source node s that for every node v in the graph contains the terminal set $\{s, v\}$. Then an (α, β) group spanner for this instance simultaneously approximates the shortest path distance from s to v for every v to within a factor of β, and has size no more than α times the size of the minimum spanning tree in the graph. However, while our approach only guarantees $\beta = O(\log n)$ for $g = n$ commodities, it is possible to obtain an $(O(1/\epsilon), 1 + \epsilon)$ approximation for any $\epsilon > 0$ [6, 18].

2 Problem Definition

In this section, we formally define Network Design with Coverage Costs. We are given a graph $G = (V, E)$ with costs c_e on edges, a universe Π of packets, and g commodities with terminal sets $X_1, \ldots, X_g \subseteq V$. The demand set of terminal set X_j is denoted $D_j \subseteq \Pi$, and we denote the collection of all demand sets as $\mathcal{D}$. A solution consists of a collection of g Steiner trees $\mathcal{T} = \{T_1, \ldots, T_g\}$ where T_j is a Steiner tree spanning terminal set X_j. The trees specify how packets are to be routed over the edges: the packets of demand D_j are routed over edges of T_j. For a solution $\mathcal{T}$, the load on edge e is $\ell_e(\mathcal{T}) = |\bigcup_{i:e \in T_i} D_i|$, i.e. the total number of distinct packets being routed over edge e. More generally, we can consider a setting in which packets have weights and we define the load on an edge to be the total weight of all of the distinct packets that an edge carries. The performance and running times of both of our algorithms are independent of the number of distinct packets, so we may assume without loss of generality that all packets have unit weight. Our goal is to find a solution $\mathcal{T}$ so as to minimize the total cost $\sum_{e \in E} c_e \ell_e(\mathcal{T})$.

We now describe the two special cases of network design with coverage costs that we study. In the following, for a subgraph H, we write $c(H)$ for the total cost of edges in H, i.e. $c(H) := \sum_{e \in H} c_e$.

Laminar demands. In this setting, the collection of demand sets is laminar: for any $D, D' \in \mathcal{D}$, $D \cap D' \neq \emptyset$ implies either $D \subseteq D'$ or $D' \subseteq D$. In this case we can transform our objective into a simpler form where the cost of each edge is charged to a collection of *disjoint* demand sets. In particular, given a solution $\mathcal{T}$, for an edge e consider the demand sets D that are maximal among the collection $\{D_j : e \in T_j\}$ of demand sets that this edge carries. Because of laminarity, these maximal demand sets are disjoint, and so the load on the edge is simply the sum of the sizes of these demand sets. Accordingly, let us define $H_D(\mathcal{T})$ to be the set of edges e such that D is a maximal set in $\{D_j : e \in T_j\}$. The packet set D will contribute to the load on these edges. Then we can write the total cost of the solution $\mathcal{T}$ as

$$\ell(\mathcal{T}) = \sum_e c_e \ell_e(\mathcal{T}) = \sum_e \sum_{D: H_D(\mathcal{T}) \ni e} c_e |D| = \sum_D |D| \sum_{e \in H_D(\mathcal{T})} c_e = \sum_D |D| c(H_D(\mathcal{T})).$$

Further note that in a feasible solution $\mathcal{T}$, for each commodity j, the subgraph $\bigcup_{D \supseteq D_j} H_D(\mathcal{T})$ contains the tree T_j and therefore spans the terminal set X_j. Therefore, instead of specifying a Steiner tree for each terminal set, it suffices to specify a forest H_D for each demand set D such that each terminal set X_j is connected in $\bigcup_{D \supseteq D_j} H_D$.

Sunflower demands. In this setting, there is a special set of packets $P \subseteq \Pi$ such that for all $i \neq j$, we have $D_i \cap D_j = P$. In other words, $D_j = P \cup P_j$ with $P_i \cap P_j = \emptyset$ for all $i \neq j$. We can again transform our objective into a simpler form. For a routing solution $\mathcal{T} = \{T_1, T_2, \ldots, T_g\}$, let H denote the subgraph obtained by taking the union of the T_js. Observe that H is a Steiner forest for $X_1, \ldots, X_g$. We have to route P over H, since all terminal sets demand P, and P_j over T_j. Thus the cost of the routing solution can be expressed as $\ell(\mathcal{T}) = |P| c(H) + \sum_j |P_j| c(T_j)$.

We will now describe a lower bound on the cost of the optimal solution in this setting. For a vertex set X and subgraph H, let $\mathrm{St}_H(X)$ denote the cost of an optimal (i.e., minimum cost) Steiner tree over X in H. Let $\mathcal{T}^* = \{T_1^*, T_2^*, \ldots, T_g^*\}$ be an optimal routing solution to the given instance and let $H^* = \bigcup_j T_j^*$. Suppose F^* is an optimal Steiner forest for $X_1, \ldots, X_g$. Since H^* is a Steiner forest for $X_1, \ldots, X_g$ and T_j^* is a Steiner tree for X_j, we have $c(H^*) \geq c(F^*)$ and $c(T_j^*) \geq \mathrm{St}_G(X_j)$. Therefore the optimal routing-solution cost can be bounded as $\ell(\mathcal{T}^*) \geq |P|\ c\,(F^*) + \sum_j |P_j|\ \mathrm{St}_G(X_j)$.

Group spanners. For a graph $G = (V, E)$ with cost c_e on edges and g terminal sets $X_1, \ldots, X_g \subseteq V$, we say that subgraph H is an (α, β) *group spanner* if $c(H) \leq \alpha c(F^*)$ and $\mathrm{St}_H(X_j) \leq \beta\, \mathrm{St}_G(X_j)$ for all j. Here F^* denotes an optimal Steiner forest for $X_1, \ldots, X_g$ in G. Note that a group spanner generalizes the notion of a spanner since the latter asks for a sparse spanning subgraph H such that for every pair of vertices (u, v) we have β stretch: $d_H(u, v) \leq \beta d_G(u, v)$. Here $d_H(u, v)$ (respectively, $d_G(u, v)$) denotes the distance, with edge lengths c_e, between vertices u and v in H (respectively, G).

The following lemma shows that a good group spanner implies an approximation for the sunflower demands setting.

▶ **Lemma 1.** *Given an (α, β) group spanner H for graph G and terminal sets $X_1, X_2, \ldots, X_g$, we can obtain an $\alpha + 2\beta$ approximation for any sunflower demands instance defined over G and X_js.*

Proof. For all j, let H_j be the Steiner trees over X_j in H obtained via the MST heuristic [31]. We set $\{H_1, H_2, \dots, H_g\}$ as the routing solution for the sunflower demand instance. The cost of this solution is no more than $|P|c(H) + \sum_j |P_j|c(H_j)$. Recall that the optimal routing-solution cost for sunflower demand instance is at least $|P|\ c(F^*) + \sum_j |P_j|\ \mathrm{St}_G(X_j)$. Therefore, using the fact that H is an (α, β) group spanner and $c(H_j) \le 2\,\mathrm{St}_H(X_j)$ (guarantee of the MST heuristic) we get the desired claim. ◀

Note that using group spanners we get an oblivious approximation in the sense that the construction uses only the knowledge of the underlying graph and the terminal sets but not the demand sets.

In Section 4 we consider unweighted graphs with terminal sets that satisfy $V = \bigcup_j X_j$. We develop an algorithm that obtains a $(14, O(\log g))$ group spanner for such an instance, and so by Lemma 1 gives an $O(\log g)$ approximation to the sunflower demands setting over the instance (see Theorem 10).

3 A 2-approximation for the laminar demands setting

Recall that in the laminar demands setting, for all $D, D' \in \mathcal{D}$ with $D \cap D' \ne \emptyset$, we have $D \subseteq D'$ or $D' \subseteq D$. As established in Section 2, in order to obtain a feasible solution in this setting, it suffices to specify a forest H_D for each demand set D such that each terminal set X_j is connected in $\bigcup_{D \supseteq D_j} H_D$. The cost of the corresponding routing is $\sum_D |D|c(H_D(\mathcal{T}))$.

Our algorithm for the laminar demands case is an extension of the Goemans-Williamson primal-dual algorithm for the Steiner Forest Problem [13]. We begin by defining the primal and dual linear programs.

In the linear program below, the variable $x_{e,D}$ denotes whether $e \in H_D$. We denote by $\delta(S)$ the set of edges crossing a cut $S \subseteq V$, and by $\mathcal{S}_D$ the collection of cuts $S \subseteq V$ that separates a terminal set X_j with $D_j \supseteq D$. The cut constraints require that each terminal set X_j is connected by $\bigcup_{D \supseteq D_j} H_D$.

$$\begin{array}{lll} \text{minimize} & \displaystyle\sum_{e, D \in \mathcal{D}} x_{e,D} \cdot |D| c_e & \\ \text{subject to} & \displaystyle\sum_{D' \supseteq D} \sum_{e \in \delta(S)} x_{e,D'} \ge 1 & \forall D \in \mathcal{D}, S \in \mathcal{S}_D \end{array}$$

The corresponding dual linear program is as follows.

$$\begin{array}{lll} \text{maximize} & \displaystyle\sum_{D \in \mathcal{D}, S \in \mathcal{S}_D} y_{D,S} & \\ \text{subject to} & \displaystyle\sum_{D' \subseteq D} \sum_{S \in \mathcal{S}_{D'} : e \in \delta(S)} y_{D',S} \le |D| c_e & \forall e, D \in \mathcal{D} \end{array}$$

3.1 Algorithm

The algorithm starts with a dual ascent stage in which it adds edges to forests $\{F_D\}_{D \in \mathcal{D}}$, and ends with a pruning stage. In the following discussion, for a demand set $D \in \mathcal{D}$ we say that $S \in \mathcal{S}_D$ is a *D-unsatisfied cut* if $(\bigcup_{D' \supseteq D} F_{D'}) \cap \delta(S) = \emptyset$. We also say that an edge e is *D-tight* if

$$\sum_{D' \subseteq D} \sum_{S \in \mathcal{S}_{D'} : e \in \delta(S)} y_{D',S} = |D| c_e.$$

In the dual ascent stage, the algorithm raises duals in phases, one per demand set $D \in \mathcal{D}$ in order of increasing size. In phase D, while there exists a D-unsatisfied cut it alternates between raising duals of the minimal D-unsatisfied cuts and adding D-tight edges to F_D. We say that S is an *active set* in the current iteration of the inner while loop if it is a minimal D-unsatisfied cut. The algorithm ensures that at the end of phase D, the edges F_D are paid for by the dual and F_D is a Steiner forest for terminal sets whose demand set contains D. In the pruning stage, the algorithm processes the demand sets in order of decreasing size and removes unnecessary edges from $\{F_D\}_{D\in\mathcal{D}}$ and returns $\{H_D\}_{D\in\mathcal{D}}$.

Algorithm 1 Primal-Dual Algorithm for Laminar Buy-at-Bulk

```
1: Initialize $F_D \leftarrow \emptyset$ for all $D \in \mathcal{D}$ and $y_{D,S} \leftarrow 0$ for all $D \in \mathcal{D}, S \subseteq V$.
2: (Dual ascent stage)
3: for $D \in \mathcal{D}$ in increasing order of size do
4:     (Start of phase $D$)
5:     while there exists a $D$-unsatisfied cut do
6:         Simultaneously raise $y_{D,S}$ for active sets $S$ until some edge $e$ goes $D$-tight.
7:         $F_D \leftarrow F_D + e$.
8:     end while
9:     (End of phase $D$)
10: end for
11: (End of dual ascent stage)
12: (Pruning stage)
13: $H_D \leftarrow F_D$ for all $D \in \mathcal{D}$.
14: for $D \in \mathcal{D}$ in decreasing order of size do
15:     for $e \in H_D$ do
16:         if $(H_D - e) \cup \bigcup_{D' \supsetneq D} H_{D'}$ is a Steiner forest for terminal sets with demand set $D$
            then
17:             $H_D \leftarrow H_D - e$.
18:         end if
19:     end for
20: end for
21: (End of pruning stage)
22: return $\{H_D\}_D$
```

The following lemma implies that we can efficiently find active sets.

▶ **Lemma 2.** *In any iteration in phase D, a set S is active if and only if it is a component of F_D and it separates a terminal set whose demand set contains D.*

Proof. Let S be an active set. By definition, S is a minimal cut in $\mathcal{S}_D$ such that $\bigcup_{D' \supseteq D} F_{D'} \cap \delta(S) = \emptyset$. Since $S \in \mathcal{S}_D$, it separates a terminal set whose demand set contains D. The algorithm processes the demand sets in increasing order of size, so we have $F_{D'} = \emptyset$ for $D' \supsetneq D$ and thus $F_D \cap \delta(S) = \emptyset$. This implies that $S \cap C = \emptyset$ or $S \cap C \supseteq C$ for every connected component C of F_D and so S is a superset of a union of connected components of F_D. By minimality, we have that S is a connected component of F_D.

For the converse, consider a connected component S' of F_D that separates a terminal set whose demand set contains D. By definition, we have $S' \in \mathcal{S}_D$. Since S' is a connected component of F_D and $F_{D'} = \emptyset$ for $D' \supsetneq D$, it is a minimal set in $\mathcal{S}_D$ such that $\bigcup_{D' \supseteq D} F_{D'} \cap \delta(S) = \emptyset$. Therefore S' is an active set. ◀

3.2 Analysis

Our analysis follows along the lines of the analysis for the Goemans-Williamson algorithm. We first establish that the primal and dual solutions generated by the algorithm are feasible.

▶ **Lemma 3.** *The primal solution $\{H_D\}_{D\in\mathcal{D}}$ and the dual solution $\{y_{D,S}\}_{D\in\mathcal{D},S\subseteq V}$ are feasible.*

Proof. We first prove that the primal solution is feasible. Consider an iteration during the pruning stage. We say that terminal set X_j is *H-disconnected* if it is disconnected with respect to edge set $\bigcup_{D\supseteq D_j} H_D$ and *H-connected* otherwise. We will show that all terminal sets are H-connected in all iterations of the pruning stage.

Observe that at the end of phase D, there are no D-unsatisfied cuts and $F_{D'} = \emptyset$ for $D' \supsetneq D$. Thus, all terminal sets with demand set D are connected with respect to edge set F_D. At the beginning of the pruning stage, we have $H_D = F_D$ for all $D \in \mathcal{D}$, and so all terminal sets are H-connected. Consider an iteration in which the algorithm deletes an edge e from H_D. By definition of H-disconnected, this can only cause a terminal set with demand set $D' \subseteq D$ to be H-disconnected. However, the algorithm will not delete e if it causes a terminal set with demand set D to be H-disconnected. Now consider a demand set $D' \subsetneq D$. Since $|D'| \leq |D|$, we still have $H_{D'} = F_{D'}$ so all terminal sets with demand set D' are H-connected. Thus, all terminal sets are H-connected throughout the pruning stage and so $\{H_D\}_{D\in\mathcal{D}}$ is a feasible primal solution.

The dual solution is feasible since the algorithm explicitly ensures that the dual variables in a tight constraint are not raised. ◀

Next, we show that in each phase D of the dual raising stage, the current active sets has average degree with respect to edges $\bigcup_{D'\supseteq D} H_{D'}$ (formally defined below) at most 2 in every iteration. This in turn implies that the primal solution has cost at most twice the total dual value. Since the dual is feasible, we have that the algorithm gives a 2-approximation. We bound the average degree of active sets by showing that $\bigcup_{D'\supseteq D} H_{D'}$ is a forest and that no inactive set has degree 1.

▶ **Lemma 4.** *For all $D \in \mathcal{D}$, we have that $\bigcup_{D'\supseteq D} H_{D'}$ is a forest.*

Proof. Suppose, towards a contradiction, that the statement is false. Let D be a maximal demand set such that $\bigcup_{D'\supseteq D} H_{D'}$ contains a cycle C. By maximality, there exists $e \in C \cap H_D$. Since e is in a cycle in $\bigcup_{D'\supseteq D} H_D$, we have that $(H_D - e) \cup \bigcup_{D'\supsetneq D} H_{D'}$ is still a Steiner forest for terminal sets with demand set D. Thus, the algorithm would have removed e from H_D and so we have a contradiction. ◀

For a subset of edges $E' \subseteq E$, let $\deg_{E'}(S) = |\delta(S) \cap E'|$ denote the number of edges in E' exiting S.

▶ **Lemma 5.** *Consider an iteration in phase D of the dual raising stage. Let S be a connected component of F_D in this iteration. If $S \notin \mathcal{S}_D$, then $\sum_{D'\supseteq D} \deg_{H_{D'}}(S) \neq 1$.*

*Proof of x*We prove the contrapositive. Suppose $\sum_{D'\supseteq D} \deg_{H_{D'}}(S) = 1$. Let e and $A \supseteq D$ be the unique edge and demand set, respectively, such that $e \in H_A \cap \delta(S)$. Since the algorithm did not delete e from H_A and $\bigcup_{D'\supseteq A} H_{D'}$ is acyclic by Lemma 4, there exists X_j with $D_j = A$ and $u, v \in X_j$ such that e is on the unique $u - v$ path in $\bigcup_{D'\supseteq A} H_{D'}$. Since $\sum_{D'\supseteq D} \deg_{H_{D'}}(S) = 1$, the path crosses S exactly once. Thus, we have that S separates u, v and so $S \in \mathcal{S}_A$. By definition of $\mathcal{S}_D$, we have $\mathcal{S}_A \subseteq \mathcal{S}_D$ and this completes the proof of the lemma. ◀

We are now ready to prove that the primal solution has cost at most twice the dual value.

▶ **Lemma 6.** $\sum_D \sum_{e \in H_D} |D| c_e \leq 2 \sum_{D,S} y_{D,S}$.

Proof. Using the fact that we only add tight edges, we have

$$\begin{aligned}
\sum_{D} \sum_{e \in H_D} |D| c_e &= \sum_{D} \sum_{e \in H_D} \left(\sum_{D' \subseteq D} \sum_{S \in \mathcal{S}_{D'} : e \in \delta(S)} y_{D',S} \right) \\
&= \sum_{D'} \sum_{S \in \mathcal{S}_{D'}} y_{D',S} \left(\sum_{D \supseteq D'} \sum_{e \in \delta(S) \cap H_D} 1 \right) \\
&= \sum_{D'} \sum_{S \in \mathcal{S}_{D'}} y_{D',S} \left(\sum_{D \supseteq D'} \deg_{H_D}(S) \right) \\
&= \sum_{D'} \sum_{S \in \mathcal{S}_{D'}} y_{D',S} \deg_{\bigcup_{D \supseteq D'} H_D}(S).
\end{aligned}$$

The second equality is obtained by rearranging, and the last follows from the fact that each edge is in H_D for at most one $D \supseteq D'$.

Suppose that in an iteration in phase D', the dual for each active set is raised by Δ. This implies $\sum_{S \in \mathcal{S}_{D'}} y_{D',S} \deg_{\bigcup_{D \supseteq D'} H_D}(S)$ increases by $\Delta \cdot \sum_{S \text{ active}} \deg_{\bigcup_{D \supseteq D'} H_D}(S)$, and $\sum_{D,S} y_{D,S}$ increases by $\Delta \cdot \#$ active sets. So it suffices to prove that in each phase D' and in each iteration within the phase, the average degree of active sets is at most 2:

$$\sum_{S \text{ active}} \deg_{\bigcup_{D \supseteq D'} H_D}(S) \leq 2 \cdot \# \text{ active sets.}$$

Fix an iteration in phase D'. Note that each active set corresponds to some connected component of $F_{D'}$ by Lemma 2. Let G' be a graph whose nodes are connected components of $F_{D'}$ and whose edge set is $\bigcup_{D \supseteq D'} H_D$. The degree of a node in G' is equal to the degree of the corresponding set with respect to edge set $\bigcup_{D \supseteq D'} H_D$. Let us say that a node of G' corresponding to an active set is an *active node*, and that any other node is *inactive*. We want to show that the average degree of active nodes in G' is at most 2. Suppose we remove all isolated nodes from G'. In the resulting graph, by Lemma 5 the degree of each inactive node is at least 2, and by Lemma 4 the average degree is at most 2. So the claim follows. ◀

Lemmas 3 and 6 gives us the following theorem.

▶ **Theorem 7.** *Algorithm 1 is a 2-approximation for network design with coverage costs in the laminar demands setting.*

4 A logarithmic approximation for the sunflower demands setting

We now consider the sunflower demands setting. First we note that a polylogarithmic hardness of approximation follows by an approximation-preserving reduction from a special case of the buy-at-bulk problem called the *single-cable buy-at-bulk problem.*

In the single-cable buy-at-bulk problem, we are given a graph $G = (V, E)$ with costs c_e on edges, a load function $f(x) = L + x$ if $x > 0$ and $f(0) = 0$, and g terminal pairs (s_i, t_i). The goal is to find routes R_i for each terminal pair minimizing the cost $\sum_e f(|\{i : R_i \ni e\}|) \cdot c_e$.

Andrews and Zhang [2] showed that the single-cable buy-at-bulk problem has no $O(\log^{\frac{1}{4}-\gamma} g)$-approximation for any constant γ under standard complexity-theoretic assumptions.[3]

Observe that we can reinterpret any instance of this problem as an instance of network design with sunflower demands over the same graph as follows: let P denote a set of L "common" packets, and for each i, P_i denote a unique singleton packet; for each terminal pair (s_i, t_i) we have a group $X_i = \{s_i, t_i\}$ with demand $P \cup P_i$. Then any solution for the former problem is also a solution for the latter with the same cost and vice versa. Thus, we get the following hardness result.

▶ **Theorem 8.** *Network design with coverage costs in the sunflower demands setting does not admit an $O(\log^{\frac{1}{4}-\gamma} g)$-approximation for any constant γ unless* $\mathrm{NP} \subseteq \mathrm{ZPTIME}(n^{\mathrm{polylog}\, n})$.

Next we prove the main technical result of this section which says that we can find a group spanner of linear size with stretch $O(\log g)$.

▶ **Lemma 9.** *Given an unweighted graph $G = (V, E)$ ($c_e = 1$ for all $e \in G$) and terminal sets $X_1, \dots, X_g$ such that $V = \bigcup_j X_j$, we can construct in polynomial time a $(14, 4\log g)$ group spanner.*

Before we prove Lemma 9, we observe that, together with Lemma 1, it implies the following result for unweighted instances of the sunflower demands setting with vertex set $V = \bigcup_j X_j$.

▶ **Theorem 10.** *Network design with coverage costs in the sunflower demands setting admits an $O(\log g)$ approximation over unweighted graphs with vertex set $V = \bigcup_j X_j$.*

In the remainder of the section we will focus on unweighted graphs and write $|H|$ to denote the cost (i.e., the number of edges) of subgraph H. Let us recall some notation: for a subgraph H, $\mathrm{St}_H(X)$ denotes the cost of an optimal (i.e., minimum cost) Steiner tree over vertex set X in H, and $d_H(u, v)$ denotes the distance between vertices u, v in H. Let T denote a minimum spanning tree of the given graph G.

Now we prove Lemma 9. To that end we consider *uniform* group spanner instances where the following holds for all j: for all strict subsets S of X_j, there exists an edge $(x, y) \in E$ such that $x \in S, y \in X_j \setminus S$. In other words, there exists an optimal Steiner tree for each X_j with no Steiner vertices and it is easy to find.

Next we show that in order to establish Lemma 9 it suffices to solve uniform instances. We can transform any given group spanner instance over an unweighted graph G with $V = \bigcup_j X_j$ into a uniform instance as follows: add to X_j all Steiner vertices in the 2-approximate Steiner tree given by the MST heuristic [31] applied over X_j in G and let X'_j be the resulting set. Since X'_j is the set of all vertices of a Steiner tree, the group spanner instance with terminal sets $X'_1, \dots, X'_g$ is a uniform one.

Say we obtain subgraph H after solving the above uniform instance and H satisfies $\mathrm{St}_H(X'_j) \leq \beta\, \mathrm{St}_G(X'_j)$ for all j and $|H| \leq \alpha |T|$. We show that H is in fact a $(2\alpha, 2\beta)$ group spanner for the original instance. The MST heuristic guarantees that $\mathrm{St}_G(X'_j) \leq 2\, \mathrm{St}_G(X_j)$; which implies $\mathrm{St}_H(X_j) \leq 2\beta\, \mathrm{St}_G(X_j)$. Finally, let F^* denote an optimal Steiner forest for $X_1, \dots, X_g$ in G. In an unweighted instance, we have that $|F^*| \geq |T|/2$. This is because $V = \bigcup_j X_j$ and each component of the forest has at least one edge[4] so $|F^*| \geq |V|/2 \geq |T|/2$. Since, $|H| \leq \alpha |T|$ we get the cost guarantee, $|H| \leq 2\alpha |F^*|$.

[3] While [2] considers a different problem, it was remarked in [3] that the construction can be used for single-cable buy-at-bulk.

[4] We assume without loss of generality that $|X_j| \geq 2$ for all j

This implies that to prove Lemma 9 we only need to solve uniform group spanner instances. In the remainder of this section, we focus on uniform instances and for ease of exposition write X_j in place of X'_j.

▶ **Lemma 11.** *Given any uniform group spanner instance with terminal sets X_j, there exists a subset of edges A of size $|A| \leq 6|T|$ such that for $H := A \cup T$ we have $\mathrm{St}_H(X_j) \leq (2\log g)\,\mathrm{St}_G(X_j)$ for all j.*

Since $|H| = |A| + |T| \leq 7|T|$ and $\mathrm{St}_H(X_j) \leq (2\log g)\,\mathrm{St}_G(X_j)$, we get that H is a $(14, 4\log g)$ group spanner that satisfies the desired bounds in Lemma 9.

We now move on to present a constructive proof of Lemma 11. We assume that terminals of X_j are ordered $x_{j,1}, x_{j,2}, \ldots$ such that for $i > 1$, there exists an edge $(x_{j,i}, x_{j,k}) \in E$ for some $k < i$; we call this edge a *satisfying edge* for $x_{j,i}$. For ease of notation, we drop the indices when they do not matter and write (x, y) to denote x's satisfying edge. Note that such an ordering always exists, e.g. a preordering of the (uniform) Steiner tree over X_j with any root. We say that a terminal $x_{j,i} \in X_j$ is *unsatisfied*[5] in a spanning subgraph H if $d_H(x_{j,i}, \{x_{j,1} \ldots, x_{j,i-1}\}) > 2\log g$. Note that a single vertex may correspond to multiple satisfied/unsatisfied terminals of different groups. The following fact implies that subgraphs in which all terminals are satisfied are group spanners with $\beta = 2\log g$.

▶ Fact 1. If H is a spanning subgraph such that $d_H(x_{j,i}, \{x_{j,1} \ldots, x_{j,i-1}\}) \leq 2\log g$ for all $i > 1$, then there exists a Steiner tree for X_j in H with total size at most $(2\log g)\,\mathrm{St}_G(X_j)$.

Our algorithm starts with the MST T and adds satisfying edges to it in order to construct H. In order to bound the cost of these edges, the algorithm maintains an arc set E' defined over the vertex set V. Let G' denote the directed graph (V, E'). At the beginning of the algorithm, E' is empty. We use *arcs* to refer to directed edges in E' and simply *edges* for edges in E. Our algorithm works in two phases. In the first phase, for each unsatisfied terminal, the algorithm adds its satisfying edge only if we can add an oriented copy of it to E' and modify nearby arcs in E' such that the out-degree of every node is at most 2. The main lemma is that the number of unsatisfied terminals at the end of this phase is at most $|V|$, and so we can simply add their satisfying edges in the second phase. We use the following notation for the algorithm: $\delta^+(x)$ denotes the number of edges of E' that are oriented away from x; $\Gamma(x) \subseteq V$ denotes the set of terminals reachable from x via a directed path in E' of length at most $\log g$.

At the end of the algorithm every vertex is satisfied. Fact 1 then implies that $H = T \cup A_1 \cup A_2$ is a group spanner with $\beta = 2\log g$. So we only need to bound the sizes of A_1 and A_2. Since there is a one-to-one correspondence between edges in A_1 and arcs in E', the following lemma implies that $|A_1| = |E'| \leq 2|V|$.

▶ **Lemma 12.** *We have $\delta^+(x) \leq 2$ for all $x \in V$.*

Proof. We prove the lemma by induction on the iterations of the algorithm. The base case ($E' = \emptyset$) is trivial. The interesting case is when $\delta^+(x) = 2$ at the beginning of the iteration and the algorithm adds (x, y) to E' oriented from x to y. At this point, we have $\delta^+(x) = 3$, $\delta^+(z) \leq 1$ and all other terminals on the $x - z$ path have out-degree at most 2 by the inductive hypothesis. When the algorithm flips the arcs on the path, it decrements $\delta^+(x)$ by 1, increments $\delta^+(z)$ by 1 and does not affect the out-degrees of other terminals on the path. This proves the lemma. ◀

[5] We define the lowest indexed vertex $x_{j,1}$ to be always satisfied.

Algorithm 2 Algorithm for uniform graph spanner instances

```
1: (Phase 1)
2: $E', A_1, A_2 \leftarrow \emptyset$
3: while there exists $x$ that is unsatisfied in $T \cup A_1$ and $z \in \Gamma(x)$ such that $\delta^+(z) \leq 1$ do
4:     Add $x$'s satisfying edge $(x, y)$ to $E'$ oriented from $x$ to $y$
5:     Add $(x, y)$ to $A_1$
6:     if $\delta^+(x) > 2$ then
7:         Flip directions of arcs in $G'$ along $x - z$ path
8:     end if
9: end while
10: (Phase 2)
11: For every $x$ unsatisfied in $T \cup A_1$, add its satisfying edge $(x, y)$ to $A_2$
12: return $A = A_1 \cup A_2$
```

Next we bound $|A_2|$.

▶ **Lemma 13.** $|A_2| \leq |V|$.

Proof. First we prove that, even if we ignore edge directions, the length of the smallest cycle (i.e. girth) in E' is at least $\log g$. Assume, towards a contradiction, that there is an undirected cycle of length $k \leq \log g$ in E'. Let (x, y) be the last arc added in the cycle. Before the algorithm added it, there is a path from x to y of length $k - 1$ in A corresponding to the other arcs in the cycle. This contradicts the condition for adding (x, y); in particular, x is not unsatisfied.

Let $U = \{x_{j,i} : x_{j,i} \text{ unsatisfied in } T \cup A_1\}$. For $x_{j,i} \in U$, we have $\delta^+(z) = 2$ for all $z \in \Gamma(x_{j,i})$ since otherwise we would have added its satisfying edge in phase 1. Since the girth of E' is at least $\log g$, we have a full binary tree of depth $\log g$ rooted at $x_{j,i}$ in E'. This implies $|\Gamma(x_{j,i})| \geq g$. Furthermore, for any $x_{j,i}, x_{j,k} \in U$ with $i > k$, we have $\Gamma(x_{j,i}) \cap \Gamma(x_{j,k}) = \emptyset$ because otherwise $d_{T \cup A_1}(x_{j,i}, x_{j,k}) \leq 2 \log g$ and $x_{j,i}$ would not have been unsatisfied in $T \cup A_1$. Therefore any terminal can belong to at most one $\Gamma(x_{j,i})$ per j, giving us $\sum_{x_{j,i} \in U} |\Gamma(x_{j,i})| \leq g|V|$. Hence we get the desired bound: $|V| \geq \sum_{x_{j,i} \in U} |\Gamma(x_{j,i})|/g \geq g|U|/g = |U| = |A_2|$. ◀

Lemmas 12 and 13 imply that $|A_1| + |A_2| \leq 3|V|$. Furthermore, the algorithm ensures that all the terminals are satisfied in $T \cup A_1 \cup A_2$. Together with Fact 1, we get Lemma 11.

References

1 Ashok Anand, Vyas Sekar, and Aditya Akella. SmartRE: an architecture for coordinated network-wide redundancy elimination. In *ACM SIGCOMM*, 2009.

2 M. Andrews and L. Zhang. Bounds on fiber minimization in optical networks with fixed fiber capacity. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 1, pages 409–419 vol. 1, March 2005.

3 Matthew Andrews. Hardness of buy-at-bulk network design. In *Foundations of Computer Science, 2004. Proceedings. 45th Annual IEEE Symposium on*, pages 115–124. IEEE, 2004.

4 Matthew Andrews and Lisa Zhang. Approximation algorithms for access network design. *Algorithmica*, 34(2):197–215, 2002.

5 Baruch Awerbuch and Yossi Azar. Buy-at-bulk network design. In *Proceedings of the 38th IEEE Symposium on Foundations of Computer Science*, pages 542–547, 1997.

6 Baruch Awerbuch, Alan Baratz, and David Peleg. Cost-sensitive analysis of communication protocols. In *Proceedings of the ninth annual ACM symposium on Principles of distributed computing*, PODC'90, pages 177–187, New York, NY, USA, 1990. ACM.

7 Siddharth Barman and Shuchi Chawla. Traffic-redundancy aware network design. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'12, pages 1487–1498. SIAM, 2012.

8 Y. Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, FOCS'96, pages 184–193, Washington, DC, USA, 1996. IEEE Computer Society.

9 BlueCoat: WAN Optimization. `http://www.bluecoat.com`.

10 Michael Elkin and David Peleg. $(1+\epsilon,\beta)$-spanner constructions for general graphs. *SIAM Journal on Computing*, 33(3):608–631, 2004.

11 Paul Erdős. Extremal problems in graph theory. In *Theory of Graphs and its Applications (Proc. Sympos. Smolenice, 1963)*, pages 29–36. Publ. House Czechoslovak Acad. Sci., Prague, 1964.

12 Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 448–455, 2003.

13 M. Goemans and D. Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24(2):296–317, 1995.

14 S. Guha, A. Meyerson, and K. Munagala. Hierarchical placement and network design problems. In *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*, pages 603–612, 2000.

15 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'03, pages 606–617, 2003.

16 Anupam Gupta, Amit Kumar, and Tim Roughgarden. Simpler and better approximation algorithms for network design. In *STOC'03: Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 365–372, 2003.

17 A. Hayrapetyan, C. Swamy, and É. Tardos. Network design for information networks. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 933–942. Society for Industrial and Applied Mathematics, 2005.

18 Samir Khuller, Balaji Raghavachari, and Neal Young. Balancing minimum spanning and shortest path trees. In *Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms*, SODA'93, pages 243–250, Philadelphia, PA, USA, 1993. Society for Industrial and Applied Mathematics.

19 Amit Kumar, Anupam Gupta, and Tim 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, pages 333–344, 2002.

20 F Thomson Leighton and Ankur Moitra. Extensions and limits to vertex sparsification. In *Proceedings of the 42nd ACM symposium on Theory of computing*, pages 47–56. ACM, 2010.

21 Adam Meyerson, Kamesh Munagala, and Serge Plotkin. Cost-distance: Two metric network design. *SIAM J. Comput.*, 38(4):1648–1659, December 2008.

22 Ankur Moitra. Approximation algorithms for multicommodity-type problems with guarantees independent of the graph size. In *Foundations of Computer Science, 2009. FOCS'09. 50th Annual IEEE Symposium on*, pages 3–12. IEEE, 2009.

23 Seth Pettie. Low distortion spanners. In *Automata, Languages and Programming*, pages 78–89. Springer, 2007.

24 Riverbed Networks: WAN Optimization. `http://www.riverbed.com/us/solutions/optimization`.

25 David B. Shmoys, Chaitanya Swamy, and Retsef Levi. Facility location with service installation costs. In *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1088–1097, 2004.

26 Neil T. Spring and David Wetherall. A protocol-independent technique for eliminating redundant network traffic. In *Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication – SIGCOMM'00*, pages 87–95, Stockholm, Sweden, 2000.

27 Zoya Svitkina and Éva Tardos. Facility location with hierarchical facility costs. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 153–161, 2006.

28 Kunal Talwar. The Single-Sink Buy-at-Bulk LP Has Constant Integrality Gap. In *Proceedings of the 9th International IPCO Conference on Integer Programming and Combinatorial Optimization*, pages 475–486, 2002.

29 Mikkel Thorup and Uri Zwick. Approximate distance oracles. *J. ACM*, 52(1):1–24, January 2005.

30 Mikkel Thorup and Uri Zwick. Spanners and emulators with sublinear distance errors. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 802–809. ACM, 2006.

31 Vijay V Vazirani. *Approximation Algorithms*. Springer, 2001.