On the Approximability of Some Network Design Problems

Julia Chuzhoy[*]   Anupam Gupta[†]   Joseph (Seffi) Naor[‡]   Amitabh Sinha[§]

Abstract

Consider the following classical network design problem: a set of terminals $T = \{t_i\}$ wants to send traffic to a "root" r in an n-node graph $G = (V, E)$. Each terminal t_i sends d_i units of traffic, and enough bandwidth has to be allocated on the edges to permit this. However, bandwidth on an edge e can only be allocated in *integral* multiples of some base capacity u_e — and hence provisioning $k \times u_e$ bandwidth on edge e incurs a cost of $\lceil k \rceil$ times the cost of that edge. The objective is a minimum-cost feasible solution.

This is one of many network design problems widely studied where the bandwidth allocation being governed by side constraints: edges may only allow a subset of cables to be purchased on them, or certain quality-of-service requirements may have to be met.

In this work, we show that the above problem, and in fact, several basic problems in this general network design framework, cannot be approximated better than $\Omega(\log \log n)$ unless $\mathsf{NP} \subseteq \mathsf{DTIME}\left(n^{O(\log \log \log n)}\right)$. In particular, we show that this inapproximability threshold holds for (i) the Priority-Steiner Tree problem [7], (ii) the Cost-Distance problem [31], and the single-sink version of an even more fundamental problem, (iii) Fixed Charge Network Flow [33]. Our results provide a further breakthrough in the understanding of the level of complexity of network design problems. These are the first non-constant hardness results known for all these problems.

1 Introduction

Approximation algorithms have had much success in the area of network design, with both combinatorial and linear-programming based techniques leading to many constant-factor approximation algorithms.

Despite these successes, several basic network design problems have eluded the quest for constant-factor approximations, with the current best approximation guarantees being logarithmic or worse. Moreover, none of the previously-known hardness results precludes the possibility of constant-factor approximation algorithms for these problems. In this paper, we make progress on this front and show $\Omega(\log \log n)$-hardness results for the following problems:

Fixed-Charge Network Flow (FCNF): The FCNF problem and its many variations has been widely studied in the Operations Research community [33, 34, 6, 19]. The *single source* version considered here is a very natural and basic network design problem. The input is a graph where each edge has a *cost* c_e and a *capacity* u_e. Given a set T of terminals $\{t_i\}$ (each with a demand d_i) and a designated *root* node r, the goal is to choose a subset of edges, such that every terminal t_i can route d_i flow units to the root. The edge capacities cannot be violated, and multiple copies of each edge e can be purchased, each incurring an additional cost of c_e.

FCNF generalizes several important network design problems and hence it is of great interest to theoreticians and practitioners alike. We show that our hardness result holds for the *single source* version of FCNF.

Priority Steiner Tree: Motivated by the heterogeneity in Quality of Service requirements in multicast (video) applications [28, 42], the Priority-Steiner Tree problem was defined by Charikar et al. [7]; special cases of this problem have also been studied in the Operations Research community [10, 11, 32, 35].

In Priority-Steiner Tree, given a graph $G = (V, E)$ with edge costs c_e, a set T of terminals and a root node r, we want to find a min-cost Steiner tree $\mathfrak{T} \subseteq G$ spanning $T \cup \{r\}$. An additional constraint on $\mathfrak{T}$ is that each terminal $t \in T$ desires a certain Quality of Service (or priority) $Q(t) \in \{1, 2, \ldots, k\}$ — here 1 is the highest and k is the lowest QoS. Furthermore, each edge offers a certain QoS $Q(e) \in \{1, 2, \ldots, k\}$, and we want the path in $\mathfrak{T}$ from $t \in T$ to the root r to consist only of edges e with QoS at least as good as $Q(t)$; i.e., $Q(e) \leq Q(t)$.

[*]Laboratory for Computer Science, MIT, Cambridge, MA. Work done while the author was a graduate student at the Computer Science Department at the Technion. Email: `cjulia@csail.mit.edu`

[†]Dept. of Computer Science, Carnegie Mellon University, Pittsburgh PA 15213. Email: `anupamg@cs.cmu.edu`

[‡]Computer Science Department, Technion, Israel Institute of Technology, Haifa 32000, Israel. Research of is supported in part by the United States-Israel Binational Science Foundation Grant No. 2002-276 and by EU contract IST-1999-14084 (APPOL II). Email: `naor@cs.technion.ac.il`

[§]Ross School of Business, University of Michigan, Ann Arbor MI 48109. Work done while the author was a graduate student at the Tepper School of Business at Carnegie Mellon University, and supported in part by NSF grant CCR-0105548 and ITR grant CCR-0122581 (The ALADDIN project). Email: `amitabh@umich.edu`

A $\min\{2\ln|t|, 1.55\,k\}$-approximation for the Priority-Steiner Tree problem was given by Charikar et al. [7]; they also showed that the problem is NP-hard even when $t = V$. However, no $\Omega(1)$ approximation hardness was previously known for the problem.

Cost-Distance: This problem was introduced by Meyerson et al. [31], as a common generalization of several problems in network design and facility location. The problem is identical to FCNF, except that there are no edge capacities, and each edge e has a *length* ℓ_e; the costs and lengths of edges may be unrelated to each other. The goal is to find a Steiner tree $\mathfrak{T}$ spanning $T \cup \{r\}$ that minimizes the objective function

$$\sum_{e \in \mathfrak{T}} c_e + \sum_{i \in T} d_i\,\ell_{\mathfrak{T}}(t_i, r);$$

i.e., the sum of edge costs of the tree together with the weighted distances from the terminals to the root.

Building on the basic framework of Marathe et al. [27], Meyerson et al. [31] gave an $O(\log|T|)$ randomized approximation algorithm for the problem. This algorithm was subsequently derandomized by Chekuri et al. [8], and an $O(\log^4 n)$-competitive online algorithm was given by Meyerson [30]. The best hardness result previously known for this problem was an $\Omega(1)$-hardness via facility location.

1.1 Our Results and Related Work

We show that it is hard to approximate the above problems to better than a factor of $\Omega(\log\log n)$. In particular, the technical heart of the paper is the following theorem:

Theorem 1.1 *It is impossible to approximate the* Priority-Steiner Tree *problem in polynomial time to better than a factor of $c\log\log n$ (for some constant c) unless $NP \subseteq$* DTIME $\left(n^{O(\log\log\log n)}\right)$. *This is true even for instances of* Priority-Steiner Tree *where all edges have unit cost.*

We then give approximation-preserving reductions from the Priority-Steiner Tree problem to the FCNF and the Cost-Distance problems, which imply similar hardness results for these problems as well.

As mentioned above, these are the first non-constant inapproximability thresholds for these network design problems. In fact, all these problems are *single-sink* problems on *undirected* graphs; non-constant hardness results that were known previously made crucial use of either (actual or simulated) directedness, such as the recent work on polylogarithmic inapproximability [22] of directed Steiner and group Steiner trees, or of the fact that there were multiple sinks (and hence multiple commodities) in the system [1].

An inapproximability threshold of $\frac{5601}{5600}$ for the basic undirected Steiner tree problem was shown in [9], which immediately implies the same threshold to all the problems we consider. A 1.46 inapproximability threshold [16] for facility location extends to Cost-Distance, as shown in [31]. To the best of our knowledge, these were the only inapproximability results known for any of the problems we consider. Hence, our work represents a substantial leap in showing that several basic network design problems are unlikely to admit constant-factor approximations.

Note that two of the three problems have $O(\log n)$ approximations ([7] for Priority-Steiner Tree and [31] for Cost-Distance). Therefore, our results show the existence of an intermediate class of network design problems which are neither constant-approximable nor polylogarithmic-inapproximable. On one extreme, we have the undirected single-source network design problems with homogenous cost functions on edges such as single-source buy-at-bulk, which admit constant factor approximations. On the other extreme, incorporating further complexity to the problems leads, as one would expect, to greater degrees of inapproximability: the current inapproximability threshold for directed Steiner tree is $\Omega(\log^{2-\epsilon} n)$ [22] and for multi-commodity buy-at-bulk network design is $\Omega(\log^{\frac{1}{4}-\epsilon} n)$ [1].

Our problems therefore occupy a middle ground with logarithmic approximations (to date), but super-constant inapproximability. Indeed, the Priority-Steiner Tree problem generalizes the undirected Steiner tree problem, but can be implemented as a special case of the directed Steiner tree problem.

Network Design: There has been much research in the area of approximation algorithms for network design, with many new techniques being developed and problem areas being explored. A partial list includes [5, 2, 12, 23, 15, 39, 14, 17, 18, 21, 20, 25, 27, 29, 31, 24, 36, 37, 39, 40, 41]; see the many references therein for more pointers.

To place our results in a better context, we focus on concave-cost network design, where the cost of using an edge is a concave function of the flow on the edge. If the cost function is the same per unit length in all edges (or *uniform*), then the problem is constant-approximable in the single-sink case [18, 41] and $O(\log n)$-approximable in the multi-commodity case [5, 13], with an $\Omega(\log^{\frac{1}{4}-\epsilon} n)$-inapproximability threshold shown by Andrews [1] for the multi-commodity case. Constant approximations exist for special cases of multi-commodity concave cost network design, such as the rent-or-buy cost function [25, 21]. Cost-Distance and FCNF are special cases of single-sink concave cost network design with *non-uniform* costs on edges; for these problems, our $\Omega(\log\log n)$-inapproximability results complement the $\Omega(\log^{\frac{1}{2}-\epsilon})$-inapproximability of the multi-commodity version in [1]. We hope that our results will help in providing further understanding of the factors determining

the degree of difficulty of approximating the different types of network design problems.

Paper Outline: We achieve our goal of showing the in-approximability of these network design problems by using one of them (Priority-Steiner Tree) to encode an instance of a Set Cover problem, which itself is shown to be hard to approximate using a reduction from MAX 3SAT(5). To this end, we begin in the next section by explicitly constructing this set system instance. Subsequently, in Section 3, we show the hardness of Priority-Steiner Tree using this construction. We then show the hardness of the other two network design problems in Section 4, using the hardness of Priority-Steiner Tree.

2 Set System Construction

In our reduction, we start from a 3SAT(5) formula φ and produce an instance of Priority-Steiner Tree. As a building block for our reduction, we use the construction of Lund and Yannakakis [26] for the hardness of the set cover problem. Since we will be using some properties of this construction which are not proved explicitly in [26] (though easily follow from the paper), as well as for the sake of completeness, we provide their construction here with slight changes (which are performed for convenience reasons).

The reduction is performed from the gap version of Exact MAX 3SAT(5), which is defined as follows. We are given a CNF formula φ with n variables and $5n/3$ clauses. Each clause contains exactly 3 literals, and each variable appears in exactly 5 different clauses. The goal is to find an assignment which maximizes the number of satisfied clauses. Formula φ is called a yes-instance, if it is satisfiable, and it is called a no-instance (with respect to some $\epsilon : 0 < \epsilon < 1$), if the maximum fraction of clauses that can be satisfied simultaneously is $(1 - \epsilon)$. The following corollary of the PCP theorem was shown by [3]

Theorem 2.1 *For some constant ϵ, it is NP-hard to distinguish between the yes and the no instances of 3SAT(5).*

2.1 The Set System

We start from a 3SAT(5) formula φ and we build a set system. A useful tool for defining the construction is the Raz verifier for 3SAT(5) with $\ell = \Theta(\log \log \log n)$ repetitions. The verifier receives as an input a 3SAT(5) formula φ, and proceeds as follows. As the first step, the verifier chooses, independently at random, a sequence of ℓ clauses $C_1, \ldots, C_\ell$ from the formula and a sequence of ℓ variables $x_1, \ldots, x_\ell$, where for each $i : 1 \leq i \leq \ell$, x_i is chosen randomly and independently from the variables participating in C_i. Variable x_i is called the distinguished variable of C_i. The indices of clauses $C_1, \ldots, C_{\ell/2}$ and of variables $x_{\ell/2+1}, \ldots, x_\ell$ are sent to the first prover, while the indices of variables $x_1, \ldots, x_{\ell/2}$ and of clauses $C_{\ell/2+1}, \ldots, C_\ell$ are

sent to the second prover. Each prover answers with an assignment to all the variables that appear in its query (both as distinguished variables and as variables belonging to the query clauses). The verifier checks that for each clause C_i, $1 \leq i \leq \ell/2$, the assignment sent by the first prover satisfies the clause and that for each clause C_i, $\ell/2 + 1 \leq i \leq \ell$, the assignment sent by the second prover satisfies the clause. If this is not true, the verifier rejects. Finally, the verifier checks that for each $i : 1 \leq i \leq \ell$, the assignments of both provers to x_i are identical, and accepts or rejects accordingly.

Theorem 2.2 ([4, 3, 38]) *If φ is a yes-instance of 3SAT(5), then there is a strategy of the provers that makes the verifier always accept. If φ is a no-instance, then no matter what the strategy of the provers is, the acceptance probability of the verifier is at most $2^{-\alpha \ell}$, for some universal constant α.*

We denote by X and Y the sets of all the possible queries to provers 1 and 2 respectively (each query contains an $\ell/2$-tuple of clauses and an $\ell/2$-tuple of variables). Given a query $q \in X \cup Y$, let $A(q)$ be the set of all the possible answers to this query that satisfy all the clauses appearing in q. Note that $|X| = |Y| \leq (2n)^\ell$, and for each $q \in X \cup Y$, $|A(q)| \leq 8^\ell$. We denote by R the set of all the random strings of the verifier, so $|R| \leq (5n)^\ell$. For a random string $r \in R$, let $q_1(r)$ and $q_2(r)$ be the queries sent to the two provers when the verifier chooses r.

Given a 3SAT(5) formula φ, we build a set system using the Raz verifier defined above. The collection of sets is denoted by $\mathcal{S}$, and it contains, for each query $q \in X \cup Y$ and for each answer $a \in A(q)$, a set $S(q, a)$, i.e.,

$$\mathcal{S} = \{S(q, a) \mid q \in X \cup Y, a \in A(q)\}$$

The number of sets is bounded by $32^\ell n^\ell$. We now proceed to define the elements. Consider some random string $r \in R$, and let $x_1, \ldots, x_\ell$ be its corresponding distinguished variables. Let $\mathcal{A}$ be the set of all the possible assignments to these variables, $|\mathcal{A}| = 2^\ell$. For each $\mathcal{A}' \subseteq \mathcal{A}$, there is an element $E(r, \mathcal{A}')$, which belongs to all such sets $S(q_1(r), a)$, where a is consistent with some assignment in $\mathcal{A}'$ (i.e., the projection of a onto the distinguished variables belongs to $\mathcal{A}'$), and to all such sets $S(q_2(r), a')$, where a' is consistent with some assignment in $\overline{\mathcal{A}'}$ (i.e., the projection of a' onto the distinguished variables is not in $\mathcal{A}'$). As $|R| \leq (5n)^\ell$, and since for each $r \in R$ there are 2^{2^ℓ} elements, the total number of elements is bounded by $(5n)^\ell \cdot 2^{2^\ell}$.

This completes the description of the set system. We will need the following properties of the system.

Set sizes: Observe that for each query $q \in X$, there are $15^{\frac{\ell}{2}}$ random strings $r \in R$, such that $q = q_1(r)$. The same is true for $q \in Y$. As there are 2^{2^ℓ} elements corresponding to r, and, given some assignment $a \in A(q)$, exactly half of

these elements belong to set $S(q, a)$, the sizes of all the sets are identical and equal to $z = \frac{1}{2} \cdot 15^{\frac{\ell}{2}} \cdot 2^{2^\ell}$.

Element degrees: Given an element $e = E(r, \mathcal{A}')$, denote by d_e the number of sets in which e participates. Let $\mathcal{A}$ be the set of all the assignments to the distinguished variables corresponding to random string r. For each $a \in \mathcal{A}$, there are exactly 2^l answers of prover 1 to query $q_1(r)$ (prover 2 to query $q_2(r)$), that are consistent with a (though some of them may not satisfy all the query clauses). This is because the distinguished variables must have the same assignment as in a, and additionally there are $2 \cdot \frac{\ell}{2} = \ell$ variables that appear in query clauses and are not distinguished variables. Therefore, $d_e \leq 2^\ell \cdot |\mathcal{A}| = 2^{2\ell}$. Putting $d = \max_e \{d_e\} \leq 2^{2\ell}$, the number of elements is at least $\frac{|\mathcal{S}|z}{d}$.

Yes-Instances: Assume φ is a yes-instance. Then there is a strategy of the provers which makes the verifier always accept. We show that we can cover all the elements by at most $|X| + |Y|$ sets. For each query $q \in X \cup Y$, choose the set $S(q, a)$, where a is the answer (under the above strategy) to query q. Now consider some element $E(r, \mathcal{A})$. Let a be the correct assignment to the distinguished variables corresponding to random string r. If $a \in \mathcal{A}$, then there is a set $S(q_1(r), a')$ in the solution ($a' \in A(q_1(r))$ and a' is consistent with a), that covers this element. Otherwise, there is a set $S(q_2, a'')$ in the solution that covers this element ($a'' \in A(q_2(r))$ and a'' is consistent with a).

No-Instances: Assume that φ is a no-instance. The following theorem shows that even if we choose a "large" number of sets, there is still a substantial fraction of elements not covered by these sets.

Theorem 2.3 *Suppose we are given a family $\mathcal{S}' \subseteq \mathcal{S}$ of sets, whose size is less than $(|X| + |Y|) \cdot h$ for some integer $h > 1$, where $2^{\alpha \ell} > 64 \, h^2$. Then, at least a fraction $\frac{1}{4 \cdot 2^{8h}}$ of the elements are not covered by sets in $\mathcal{S}'$.*

Proof. Let $Q \subseteq X \cup Y$ be the subset of queries, such that for each query $q \in Q$, the number of sets $S(q, a) \in \mathcal{S}'$ is at least $4h$. By a simple averaging argument, Q contains at most a fraction $\frac{1}{4}$ of queries.

Let $R' \subset R$ be the subset of random strings, such that $q_1(r) \notin Q$ and $q_2(r) \notin Q$. As each query participates in the same number of constraints, $|R'| \geq \frac{1}{2}|R|$.

Let $R'' \subseteq R'$ be the subset of random strings r, such that there are sets $S(q_1(r), a)$ and $S(q_2(r), a')$ in $\mathcal{S}$, where a and a' are consistent. We prove the following lemma.

Lemma 2.1 $|R''| \leq \frac{1}{2}|R'|$.

Proof. Assume otherwise. We show an assignment that satisfies a large number of constraints. The assignment is as follows. Each query $q \notin Q$ chooses randomly one of its $\leq 4h$ assignments that correspond to the sets in $\mathcal{S}$. The probability that the verifier chooses a random string in

R'' is at least $\frac{1}{4}$. If this happens, then the probability that the answers of the provers are consistent is at least $\frac{1}{16h^2}$. Thus in total, the verifier accepts with probability at least $\frac{1}{64h^2} > 2^{-\alpha l}$. □

Consider some random string $r \in R' \setminus R''$ (i.e., $\mathcal{S}$ contains less than $4h$ sets from the family $\{S(q_1(r), a) \mid a \in A(q_1(r))\}$ and less than $4h$ sets from the family $\{S(q_2(r), a') \mid a' \in A(q_2(r))\}$, and no pair of sets $S(q_1(r), a)$, $S(q_2(r), a')$ where a, a' are consistent). Let $e = E(r, \mathcal{A})$ be some element and let A'_x be the subset of assignments to the distinguished variables, such that for each $a \in A'_x$, there is some set $S(x, a') \in \mathcal{S}$ where a' is consistent with a. Define A'_y similarly. Clearly, $|A'_x| \leq 4h$, $|A'_y| \leq 4h$ and $A'_x \cap A'_y = \emptyset$. Element $e = E(x, y, \mathcal{A})$ is covered by sets in $\mathcal{S}$ iff $\mathcal{A} \cap A'_x \neq \emptyset$, or $\overline{\mathcal{A}} \cap A'_y \neq \emptyset$. As for each assignment a to the distinguished variables, exactly half the sets $\mathcal{A}$ contain this assignment and half the sets $\mathcal{A}$ do not contain it, the fraction of elements corresponding to random string r which are not covered is at least $\frac{1}{2^{8h}}$.

Since $|R' \setminus R''| \geq \frac{1}{4}|R|$, the total fraction of elements which are not covered by $\mathcal{S}'$ is at least $\frac{1}{4 \cdot 2^{8h}}$. This completes the proof of Theorem 2.3. □

3 Hardness of Priority Steiner Tree

In Priority-Steiner Tree, we are given a graph $G = (V, E)$ with edge costs $c_e \in \mathbb{R}_{\geq 0}$ and edge priorities $Q(e) \in \{1, 2, \ldots, k\}$. (We will use the terms "priority" and "level" interchangeably; it is useful to remember that level 1 is the highest priority and level k is the lowest, and hence a *higher* priority corresponds to a *lower* number.) There is also a set T of terminals and a root node r, with each terminal $t \in T$ desiring a certain priority $Q(t) \in \{1, 2, \ldots, k\}$. The goal is to build a min-cost Steiner tree $\mathfrak{T}$ spanning the root r and all the terminals, such that each edge e on the path from $t \in T$ to r has priority $Q(e) \leq Q(t)$.

We now describe the reduction from the gap version of 3SAT(5) to the Priority-Steiner Tree problem. The value of k will be specified later.

3.1 The Reduction from Gap-3SAT(5)

In order to define the graph, we use the set system constructed in Section 2.1. (Recall that the collection of sets is denoted by $\mathcal{S}$). The graph consists of a root r, and of a collection of $|\mathcal{S}|$ disjoint sets of non-terminal vertices $V_1, \ldots, V_{|\mathcal{S}|}$, each set of cardinality $M + 1$ (M is a large integer to be determined later). The vertices in set V_i, $1 \leq i \leq |\mathcal{S}|$ are denoted by $v_0^i, \ldots, v_M^i$. For each $i : 1 \leq i \leq |\mathcal{S}|$, for each pair of successive vertices v_j^i, v_{j+1}^i (where $0 \leq j < M$), there are k edges connecting v_j^i and v_{j+1}^i of k different priority levels. The weights (or lengths) of all these edges are $\frac{1}{|\mathcal{S}|M}$.

Intuitively, it is convenient to view this construction as follows. For each $i : 1 \leq i \leq |\mathcal{S}|$, there is a path of length

$\frac{1}{|\mathcal{S}|}$, representing set $S_i \in \mathcal{S}$. The path is denoted by P_i and it consists of many small edges. The number of edges is denoted by M, which is a large integer to be determined later. So the length (or the cost) of each edge along path P_i is $\frac{1}{|\mathcal{S}|M}$, and the total length of all the paths is unit. Finally, in order to obtain our construction, for each path P_i, $1 \leq i \leq |\mathcal{S}|$, each edge belonging to P_i is replaced by k parallel edges of different priority levels.

In addition to the sets of edges and non-terminal vertices described above, for each priority level $j : 1 \leq j \leq k$, there is a set T_j of level j terminals and a set E_j of level j edges, which are described below. The weight of each edge in E_j is 0.

The definition of the terminal and edge sets, T_j and E_j, is recursive, and we start from the highest priority level 1. The set of terminals T_1 contains, for every element e in our set system, a priority level 1 terminal $t^1(e)$, which represents this element. In order to define the set of edges E_1, consider some path P_i, $1 \leq i \leq |\mathcal{S}|$, corresponding to set $S_i \in \mathcal{S}$. Recall that z denotes the set size and let $e_{i_1}, \ldots, e_{i_z}$ be the elements belonging to S_i. We subdivide P_i into $2z$ equal-length subpaths and denote the endpoints of these subpaths by $p_0, p_1, \ldots, p_{2z}$. Connect each one of the vertices $p_2, p_4, \ldots, p_{2z-2}$ to the root, and for each $a, 1 \leq a \leq z$, connect vertex p_{2a-1} to terminal $t^1(e_{i_a})$. All the newly added edges belong to E_1 and have priority level 1.
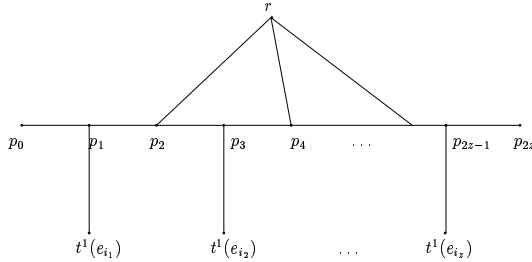

Figure 1: Level 1 construction for path P_i.

Thus, for each $i : 1 \leq i \leq \mathcal{S}$, we have divided path P_i into $2z$ subpaths. These subpaths are called level-1 subpaths of P_i. Each one of these paths is in turn divided into $2z$ equal-length subpaths, called level-2 subpaths. Generally, for each priority level $j : 1 \leq j \leq k$, path P_i is divided into $(2z)^j$ level-j subpaths.

Consider some priority level j, $1 < j \leq k$. We define the corresponding set of terminals T_j and set of edges E_j. For each set $S_i \in \mathcal{S}$ in our set system, consider the corresponding path P_i. This path is divided into $n_{j-1} = (2z)^{j-1}$ level-$(j-1)$ subpaths. We will use n_{j-1} copies of the set system, while for each $i : 1 \leq i \leq |\mathcal{S}|$ and for each $b : 1 \leq b \leq n_{j-1}$, the bth subpath of P_i represents the set S_i

in b^{th} set system.

The set T_j of terminals is defined as follows. For each $b : 1 \leq b \leq n_{j-1}$, for each element e in the set system, there is a terminal $t_b^j(e)$ in T_j. The priority level of all the terminals in T_j is j. We now proceed to define the set of level-j edges E_j. Given some $i : 1 \leq i \leq \mathcal{S}$ and $b : 1 \leq b \leq n_{j-1}$, consider the bth level-$(j - 1)$ subpath of P_i. We divide this subpath into $2z$ equal-length level-j subpath. Denote the endpoints of these subpaths by $p_0, p_1, \ldots, p_{2z}$. Connect the vertices $p_2, p_4, \ldots, p_{2z-2}$ to the root and for each $a : 1 \leq a \leq z$, connect vertex p_{2a-1} to terminal $t_b^j(e_{i_a})$. The priority levels of the newly added edges are j.

This concludes the definition of the reduction. Note that the only non-zero cost edges in our construction are the edges on paths $P_1, \ldots, P_{|\mathcal{S}|}$. Moreover, all the edges on these paths have the same cost, including parallel edges with different priority levels. So, obviously, an optimal solution can buy only level-1 edges on these paths (but will also have to use some zero-cost edges in $E_1, \ldots, E_k$ of priority levels $1, \ldots, k$). The idea is that in the yes-instance it is enough to buy a small fraction of edges on paths $P_1, \ldots, P_{|\mathcal{S}|}$. More specifically, for each set S_i belonging to the optimal set cover solution, we will buy all the level-1 edges on path P_i. On the other hand, in the case of no-instance, any solution will have to use a large fraction of edges on paths $P_1, \ldots, P_{|\mathcal{S}|}$.

We now proceed to analyze the properties of our reduction.

3.2 Yes-instance

Theorem 3.1 *Suppose φ is a Yes-instance. Then there is a solution of the corresponding priority Steiner tree problem instance whose cost is at most $\frac{|X|+|Y|}{|\mathcal{S}|}$.*

Proof. Since φ is a yes-instance, there is a family $\mathcal{S}' \subset \mathcal{S}$ of sets that cover all the elements, and $|\mathcal{S}'| = |X| + |Y|$. The solution to the priority Steiner tree problem contains, for each $S_i \in \mathcal{S}'$, all the priority-level 1 edges on path P_i and also all the edges in $E_1, \ldots, E_k$ that are adjacent to some vertex on path P_i. As $\mathcal{S}'$ constitutes a feasible set cover, for each priority level $j : 1 \leq j \leq k$, for each set cover instance $b : 1 \leq b \leq n_{j-1}$, all the terminals corresponding to the elements of this instance of set cover are connected to the root. Thus, we obtain a feasible solution whose cost is bounded by $\frac{|X|+|Y|}{|\mathcal{S}|}$. □

3.3 No-instance
In order to simplify the analysis, we assume without loss of generality that any solution uses lowest possible priority edges, i.e., if edge e of priority j is in the solution, then using priority $j+1$ edge instead of e will make the solution infeasible. We can also assume that each level-j terminal t is connected to r via a subpath of one of the

paths $P_1, P_2, \ldots, P_{|\mathcal{S}|}$ of length $\frac{1}{|\mathcal{S}|(2z)^j}$, and different terminals of the same priority level are connected via distinct subpaths. (For example, the path from terminal $t_b^j(e_{i_a})$ goes from the terminal to p_{2a-1} to p_{2a} to the root, where the segment (p_{2a-1}, p_{2a}) is a level-j subpath as described in the construction.) Therefore, for each level-j subpath p of each path P_i (where $1 \leq j \leq k, 1 \leq i \leq |\mathcal{S}|$), for each priority level $j' \leq j$, either all the edges of this priority on path p are in the solution or none of them is.

Given a solution to the priority Steiner tree problem instance, for each priority level j, $1 \leq j \leq k$, let c_j denote the cost of all the edges in the solution whose priority levels are $1, 2, \ldots, j$.

Theorem 3.2 *Suppose φ is a no-instance of 3SAT(5). Set $h = \Theta(\log \log n)$ and $\ell = \Theta(\log \log \log n)$, so that $2^{\alpha \ell} > 64h^2$ holds. Assume that for some priority level $j : 1 < j \leq k$, $c_{j-1} < \frac{h}{2|\mathcal{S}|}(|X| + |Y|)$. Then, the cost of priority level j edges in the solution is at least $\frac{1}{16 \cdot 2^{8h} \cdot 2^{2\ell}}$.*

Proof. Consider some $i : 1 \leq i \leq |\mathcal{S}|$ and $b : 1 \leq b \leq n_{j-1}$, and let p be the bth level-$(j-1)$ subpath of P_i. Recall that this subpath represents set S_i in the bth set cover instance of priority level j. We say that this set is chosen by the solution iff for some priority level $j' < j$, all the edges of p of priority level j' are in the solution (note that by our assumption, if this set is not chosen, then no edge on path p of priority levels $1, \ldots, j-1$ is in the solution).

As $c_{j-1} < \frac{h}{2|\mathcal{S}|}(|X| + |Y|)$, the total number of level-j sets that are chosen by the solution is at most $\frac{h}{2}(|X| + |Y|) \cdot (2z)^{j-1}$ (recall that the length of the path representing each such set is $\frac{1}{|\mathcal{S}| \cdot (2z)^{j-1}}$). Therefore, in at least half of the level-j set cover instances, less than $h(|X| + |Y|)$ sets are chosen. By Theorem 2.3, in each one of these instances, a fraction of at least $\frac{1}{4 \cdot 2^{8h}}$ of elements do not belong to the sets chosen by the solution. Each one of the corresponding level-j terminals must be connected to the root by a level-j subpath of one of the paths $P_1, \ldots, P_{|\mathcal{S}|}$, and these subpaths are disjoint. The cost of each such subpath is $\frac{1}{|\mathcal{S}|(2z)^j}$. As the number of elements in the set cover instance is at least $\frac{|\mathcal{S}|z}{d}$ and the number of level-j set-cover instances is $(2z)^{j-1}$, the cost of priority level j edges used by the solution must be at least:

$$\frac{(2z)^{j-1}}{2} \cdot \frac{|\mathcal{S}|z}{d} \cdot \frac{1}{4 \cdot 2^{8h}} \cdot \frac{1}{|\mathcal{S}|(2z)^j} = \frac{1}{16 \cdot 2^{8h} \cdot d}$$
$$\geq \frac{1}{16 \cdot 2^{8h} \cdot 2^{2\ell}}$$

This completes the proof of the theorem. $\square$

Corollary 3.1 *Let the number of priority levels k in our construction be $16 \cdot 2^{8h} \cdot 2^{2\ell}$. Then the solution cost is at least $\frac{h}{2|\mathcal{S}|}(|X| + |Y|)$.*

Proof. Suppose this is not true. Then for each priority level $j : 1 < j \leq k$, the conditions of Theorem 3.2 hold, and thus the cost of the priority level j edges in the solution is at least $\frac{1}{16 \cdot 2^{8h} \cdot 2^{2\ell}}$. Since this is true for each one of $k = 16 \cdot 2^{8h} \cdot 2^{2\ell}$ priority levels, it is impossible that the total cost of the solution is less than $\frac{h}{2|\mathcal{S}|}(|X| + |Y|)$. $\square$

3.4 *Setting the Parameters

It is easy to see that parameter M should be at least $(2z)^k$ and that the construction size is bounded by $N = O(|\mathcal{S}|(2z)^k)$. Recall that $k = 16 \cdot 2^{8h} \cdot 2^{2\ell}$ and that in the set cover construction, $2^{\alpha \ell} \geq 64h^2$ is required ($\alpha < 1$ is a constant). We are going to choose $h = \Theta(\log \log n)$ and $\ell = \Theta(\log \log \log n)$, so that the above inequality holds. Therefore we can bound k by $2^{O(h)}$ and

$$N \leq 32^{\ell} n^{\ell} (15^{\ell} 2^{2^{\ell}})^{2^{O(h)}} \leq n^{\ell} \cdot 2^{2^{O(h)}}$$

Since we choose $h = \Theta(\log \log n)$, we have that $N \leq n^{O(\log \log \log n)} 2^{2^{\log \log n}}$ holds, and thus $h = \Theta(\log \log N)$ as well. Observing that the yes and the no instances differ by the factor of $\frac{h}{4} = \Theta(\log \log N)$, we have proved the following theorem:

Theorem 3.3 *There is no $c \log \log n$ approximation for the priority Steiner tree problem (for some constant c), unless $NP \subseteq DTIME(n^{O(\log \log \log n)})$.*

Note that the above theorem can be extended to show an $\Omega(\log \log n)$ -hardness for the special case of priority Steiner tree problem where all the edges have unit costs, as follows. The construction remains the same, except that every edge e on paths $P_1, \ldots, P_{|\mathcal{S}|}$ is replaced by a path of $|T|$ edges, whose priority levels are the same as that of e (recall that $|T|$ is the number of terminals). The costs of all the edges are unit. It is not hard to see that the gap between the yes and the no instances is $\Omega(\log \log N')$, where $N' \leq N^2$ is the size of the new construction.

4 Hardness of Other Network Design Problems

In this section, we show that the $\Omega(\log \log n)$ hardness result of Priority Steiner can be extended to prove identical inapproximability of two popular network design problems: fixed charge network flow, and cost-distance network design.

4.1 Fixed charge network flow

The single source fixed charge network flow problem is defined on an n-node graph G with a specified root vertex r. Each node v has demand d_v and each edge e has capacity u_e and unit cost. A feasible solution specifies an integral number of copies x_e of each edge e that must be purchased so that the demand from each vertex can be simultaneously routed to the root. The total cost of the solution is therefore $\sum_e c_e x_e$, and the objective

is to find x to minimize it.[1]

Suppose we are given an instance of priority Steiner tree, where the costs of all the edges are unit. We convert it into an instance of fixed charge network flow as follows. Let k be the total number of priority levels. The underlying graph is unchanged. For each vertex v of priority i, we set the demand to be $d_v = n^{5(k-i)}$, and for each edge e of priority i the capacity is $u_e = n^{5(k-i)+2}$.

Given a solution $\mathfrak{T}_P$ for the priority Steiner tree instance, we can construct a solution $\mathfrak{T}_F$ for the fixed charge network flow instance of no greater cost, as follows. Consider some edge e. If it belongs to the solution $\mathfrak{T}_P$, then set $x_e = 1$. Otherwise, $x_e = 0$. Clearly, the costs of the two solutions are identical. It remains to show that $\mathfrak{T}_F$ is a feasible solution for the fixed charge network flow problem. Consider an edge e of priority i, and let $U(e)$ be the set of terminals whose paths to the root in $\mathfrak{T}_P$ use edge e. Clearly, the priority of all terminals in $U(e)$ are in $\{i, i+1, \ldots, k\}$. Since there are at most n terminals of each priority level, the total demand of all the vertices in $U(e)$ is at most $n \cdot n^{5(k-i)} + n \cdot n^{5(k-i-1)} + \cdots + n \cdot n^5 + n \leq n^{5(k-i)+2}$ so the capacity of edge e is sufficient to serve all nodes in $U(e)$.

The other direction is also true. Suppose we are given an optimal solution $\mathfrak{T}_F$ to the fixed charge network flow problem. We can construct a solution $\mathfrak{T}_P$ to the priority Steiner tree problem of no greater cost, as follows. $\mathfrak{T}_P$ is also defined to be identical to $\mathfrak{T}_F$, except that for every edge e with $x_e \geq 1$, we use a single edge in $\mathfrak{T}_P$. Therefore, the cost of $\mathfrak{T}_P$ is no greater that that of $\mathfrak{T}_F$. It now suffices to show the feasibility of $\mathfrak{T}_P$. First, observe that since there are at most n terminal nodes and each terminal node has a path of length at most n to the root, and since $\mathfrak{T}_F$ is an optimal solution, its cost cannot exceed n^2. Now suppose for contradiction that $\mathfrak{T}_P$ was infeasible. That is, for some terminal t of priority i, there is a cut that separates it from the root and all the edges of this cut belonging to $\mathfrak{T}_P$ have priorities in $\{i+1, i+2, \ldots, k\}$.

Note that the demand of t is n^{5i}, whereas the capacities of these edges are at most n^{5i-3}. But then in order to supply the demand of t, the number of edges in this cut that belong to $\mathfrak{T}_F$ must be at least n^3, contradicting the fact that the solution cost is at most n^2.

The next theorem follows from the above discussion and Theorem 3.3.

Theorem 4.1 *The Fixed-Charge Network Flow problem is $\Omega(\log \log n)$-hard to approximate, even in the single-source*

---

[1]Note that the general version of FCNF also incorporates an incremental cost l_e for each edge, so that the cost of sending f_e units of flow on edge e is $c_e \lceil f_e / u_e \rceil + l_e f_e$. We show that even with $l_e = 0$ our inapproximability result holds. Also, while the classical version of FCNF has $x_e \in \{0, 1\}$, this can easily be incorporated by replacing each edge with a multi-edge.

case, unless $NP \subseteq DTIME(n^{O(\log \log \log n)})$.

Note that the demands and the capacities in the above reduction could be as large as $n^{O(n)}$. However, in our construction for the priority Steiner tree problem, the number of priority levels is $\Theta(\log n)$. Therefore, we have shown $\Omega(\log \log n)$-hardness for fixed charge network flow unless $NP \subseteq DTIME(n^{O(\log n)})$, even if the demands and the capacities are given in unary.

Furthermore, note that there are only $O(\log n)$ different values of u_e used in the instance I_F. This brings our result into the realm of modern-day telecom network design, where only a few cable types are used to design networks to serve massive numbers of nodes.

4.2 Cost-distance network design An instance of the Cost-Distance network design problem is defined on n-node graphs rooted at r as follows. Each edge e has a distance l_e and a cost c_e, and each node v has demand d_v. A feasible solution consists of a tree $\mathfrak{T}_C$ spanning all nodes with positive demand and the root. Let $V(e)$ be the set of nodes whose paths to the root use edge e. The total cost of the solution is given by $c(\mathfrak{T}_C) = \sum_e \left(c_e + l_e \sum_{v \in V(e)} d_v \right)$. The objective is a minimum cost feasible solution.

Cost-Distance is also a special case of the general FCNF problem. To show the inapproximability of Cost-Distance, we use a standard reduction of single-source zero-incremental cost FCNF to Cost-Distance. Given an instance I_F of FCNF, we convert it to an instance I_C of CD by defining the length l_e of edge e to be $l_e = c_e / u_e$. The costs of edges and demands at nodes are unchanged.

Theorem 4.2 *It is not possible to approximate the Cost-Distance problem to better than an $\Omega(\log \log n)$ factor unless $NP \subseteq DTIME(n^{\log \log \log n})$.*

Proof. Let $\mathfrak{T}$ be an underlying tree rooted at r, with its counterparts in I_F and I_C denoted as $\mathfrak{T}_F$ and $\mathfrak{T}_C$ respectively. It was shown in [31, 14] (among others) that the following relationship holds between their costs: $c(\mathfrak{T}_F) \leq c(\mathfrak{T}_C) \leq 2c(\mathfrak{T}_F)$. We briefly prove this for expositionary clarity.

Consider an edge e with flow f_e. Since the tree is the same in both I_F and I_C, the flow is the same in both cases. The cost of this edge in I_F and I_C are, respectively, $c_e \lceil f_e / u_e \rceil$ and $c_e + f_e l_e$, where $l_e = c_e / u_e$. We therefore have $f_e l_e = f_e c_e / u_e \leq c_e \lceil f_e / u_e \rceil$. The claim now follows by observing that either the edge has zero flow (and hence zero cost) or positive flow (and hence cost at least c_e in I_F).

Since solutions to FCNF and Cost-Distance are within constant factors of each other, we can apply Theorem 4.1 and this theorem follows. $\square$

At this point, it is worth considering the relationship of Cost-Distance to the single-source buy-at-bulk problem,

which is known to have a constant factor approximation [18, 41]. The fundamental difference is that in the buy-at-bulk problem, edge costs and lengths are *uniform*; that is, they are proportional to edge lengths and universally available. In other words, the same set of cables is available for installation on every edge. In contrast, the FCNF problem and Cost-Distance are non-uniform, so that each edge may have its own set of available costs, lengths and capacities, with no relation whatsoever with other edges. Our results point to the fact that this non-uniformity plays a fundamental role in separating the approximability of such problems from homogenous network design problems like single-source buy-at-bulk. A similar distinction was pointed out by Andrews [1], who proved stronger inapproximability results for non-uniform multi-commodity buy-at-bulk network design.

5 Conclusions

Designing networks in practice often involves various levels of complexity and requirements, and an understanding of precisely what characteristics of the problem govern their level of approximability is critical. While our work makes some progress in this quest, several important questions remain. For instance, the gap between the approximability of Cost-Distance and Priority-Steiner Tree (both $O(\log n)$) and their inapproximability remains to be closed. The class of problems that can be modeled via FCNF-type constructions is vast, and the approximability of FCNF as defined in this paper is still open. Finally, designing networks on directed graphs presents several challenges which are as yet poorly understood.

References

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

[2] Matthew Andrews and Lisa Zhang. Approximation algorithms for access network design. *Algorithmica*, 34(2):197–215, 2002. (Preliminary version in 39th FOCS, 1998.).

[3] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, 1998.

[4] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: a new characterization of NP. *J. ACM*, 45(1):70–122, 1998.

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

[6] Robert D. Carr, Lisa Fleischer, Vitus J. Leung, and Cynthia A. Phillips. Strengthening integrality gaps for capacitated network design and covering problems. In *Symposium on Discrete Algorithms*, pages 106–115, 2000.

[7] Moses Charikar, Joseph (Seffi) Naor, and Baruch Schieber. Resource optimization in QoS multicast routing of real-time multimedia. *IEEE/ACM Transactions on Networking*, 12(2):340–348, 2004.

[8] Chandra Chekuri, Sanjeev Khanna, and Joseph (Seffi) Naor. A deterministic algorithm for the cost-distance problem. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 232–233, 2001.

[9] Andrea E.F. Clementi and Luca Trevisan. Improved non-approximability results for minimum vertex cover with density constraints. *Theoretical Computer Science*, 225(1–2):113–128, 1999.

[10] J. R. Current, C. S. Revelle, and J. L. Cohon. The hierarchical network design problem. *European Journal of Operational Reasearch*, 27:57–66, 1986.

[11] Cees Duin and Ton Volgenant. The multi-weighted Steiner tree problem. *Ann. Oper. Res.*, 33(1-4):451–469, 1991. Topological network design (Copenhagen, 1989).

[12] Guy Even, Guy Kortsarz, and Wolfgang Slany. On network design: fixed charge flows and the covering steiner problem. In *Proceedings of the 8th Scandinavian Workshop on Algorithm Theory*, volume 2368 of *Lecture Notes in Computer Science*, pages 318–329. Springer, 2002.

[13] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, pages 448–455, 2003.

[14] Naveen Garg, Rohit Khandekar, Goran Konjevod, R. Ravi, F. Sibel Salman, and Amitabh Sinha. On the integrality gap of a natural formulation of the single-sink buy-at-bulk network design formulation. In *Proceedings of the 8th Integer Programming and Combinatorial Optimization Conference*, volume 2081 of *Lecture Notes in Computer Science*, pages 170–184, 2001.

[15] Michel X. Goemans and David P. Williamson. The primal-dual method for approximation algorithms and its application to network design problems. In Dorit S. Hochbaum, editor, *Approximation Algorithms for NP-hard Problems*. PWS Publishing, 1997.

[16] Sudipto Guha and Samir Khuller. Greedy strikes back: Improved facility location algorithms. *Journal of Algorithms*, 31(1):228–248, 1999.

[17] Sudipto Guha, Adam Meyerson, and Kamesh Munagala. Hierarchical placement and network design problems. In *Proceedings of the 41th Annual IEEE Symposium on Foundations of Computer Science*, pages 603–612, 2000.

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

[19] Oktay Gunluk. A branch-and-cut algorithm for capacitated network design problems. *Mathematical Programming*, 86:17–39, 1999.

[20] Anupam Gupta, Amit Kumar, Martin Pál, and Tim Roughgarden. Approximations via cost-sharing. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, pages 606–615, 2003.

[21] 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*, pages 365–372, 2003.

[22] Eran Halperin and Robert Krauthgamer. Polylogarithmic inapproximability. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, pages 585–594, 2003.

[23] Kamal Jain. A factor 2 approximation algorithm for the generalized Steiner network problem. *Combinatorica*, 21(1):39–60, 2001. (Preliminary version in *39th FOCS*, pages 448–457, 1998).

[24] David R. Karger and Maria Minkoff. Building Steiner trees with incomplete global knowledge. In *Proceedings of the 41th Annual IEEE Symposium on Foundations of Computer Science*, pages 613–623, 2000.

[25] Amit Kumar, Anupam Gupta, and Tim Roughgarden. A constant-factor approximation algorithm for the multicommodity rent-or-buy problem. In *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science*, pages 333–342, 2002.

[26] Carsten Lund and Mihalis Yannakakis. On the hardness of approximating minimization problems. *J. Assoc. Comput. Mach.*, 41(5):960–981, 1994.

[27] Madhav V. Marathe, R. Ravi, Ravi Sundaram, S. S. Ravi, Daniel J. Rosenkrantz, and Harry B. Hunt, III. Bicriteria network design problems. *J. Algorithms*, 28(1):142–171, 1998.

[28] N. F. Maxemchuk. Video distribution on multicast networks. *IEEE J. on Selected Areas in Communications*, 15:357–372, 1997.

[29] Vardges Melkonian and Éva Tardos. Approximation algorithms for a directed network design problem. In *Integer programming and combinatorial optimization (Graz, 1999)*, volume 1610 of *Lecture Notes in Comput. Sci.*, pages 345–360. Springer, Berlin, 1999.

[30] Adam Meyerson. Online algorithms for network design. In *Proceedings of the 16th ACM Symposium on Parallelism in Algorithms and Architectures*, 2004.

[31] Adam Meyerson, Kamesh Munagala, and Serge Plotkin. Cost-distance: Two metric network design. In *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science*, pages 624–630, 2000.

[32] P. Mirchandani. The multi-tier tree problem. *INFORMS Journal on Computing*, 8:202–218, 1996.

[33] George L. Nemhauser and Laurence A. Wolsey. *Integer and Combinatorial Optimization*. 1999.

[34] Francisco Ortega and Laurence A. Wolsey. A branch-and-cut algorithm for the single-commodity, uncapacitated, fixed-charge network flow problem. *Networks*, 41(3):143–158, 2003.

[35] H. Pirkul, J. Current, and V. Nagarajan. The hierarchical network design problem: a new formulation and solution procedures. *Transportation Science*, 25:175–182, 1991.

[36] R. Ravi and F. S. Salman. Approximation algorithms for the traveling purchaser problem and its variants in network design. In *Algorithms—ESA '99 (Prague)*, volume 1643 of *Lecture Notes in Comput. Sci.*, pages 29–40. Springer, Berlin, 1999.

[37] R. Ravi and Amitabh Sinha. Integrated logistics: Approximation algorithms combining facility location and network design. In *Proceedings of the 9th Integer Programming and Combinatorial Optimization Conference*, volume 2337 of *Lecture Notes in Computer Science*, pages 212–229, 2002.

[38] Ran Raz. A parallel repetition theorem. *SIAM J. Comput.*, 27(3):763–803 (electronic), 1998.

[39] F. Sibel Salman, Joseph Cheriyan, R. Ravi, and Sairam Subramanian. Approximating the single-sink link-installation problem in network design. *SIAM Journal on Optimization*, 11(3):595–610, 2000.

[40] Chaitanya Swamy and Amit Kumar. Primal-dual algorithms for the connected facility location problem. In *Proceedings of the 5th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, volume 2462 of *Lecture Notes in Computer Science*, pages 256–269, 2002.

[41] Kunal Talwar. Single-sink buy-at-bulk LP has constant integrality gap. In *Proceedings of the 9th Integer Programming and Combinatorial Optimization Conference*, volume 2337 of *Lecture Notes in Computer Science*, pages 475–486, 2002.

[42] T. Turletti and J.-C. Bolot. Issues with multicast video distribution in heterogeneous packet networks. In *Proceedings of 6th International Workshop on Packet Video*, 1994.