On the Approximability of Some Network Design Problems

JULIA CHUZHOY
Computer Science and Artificial Intelligence Laboratory, MIT
and Department of Computer and Information Science, University of Pennsylvania
and
ANUPAM GUPTA
Computer Science Department
Carnegie Mellon University
and
JOSEPH (SEFFI) NAOR
Computer Science Department
Technion, Israel
and
AMITABH SINHA
Ross School of Business
University of Michigan

Authors' addresses: J. Chuzhoy, CSAIL MIT, Cambridge, MA 02139, and Dept. of Comp. & Inf. Sci. University of Pennsylvania, Philadelphia, PA 19104. Email: cjulia@csail.mit.edu. Work done while the author was a graduate student at the Computer Science Department at the Technion.
A. Gupta, Dept. of Computer Science, Carnegie Mellon University, Pittsburgh PA 15213. Email: anupamg@cs.cmu.edu. Research supported in part by an NSF CAREER award CCF-0448095, and by an Alfred P. Sloan Fellowship.
J. Naor, Computer Science Department, Technion, Israel Institute of Technology, Haifa 32000, Israel. Email: naor@cs.technion.ac.il. Research supported in part by the United States-Israel Binational Science Foundation Grant No. 2002-276 and by EU contract IST-1999-14084 (APPOL II).
A. Sinha, Ross School of Business, University of Michigan, Ann Arbor MI 48109. Email: amitabh@umich.edu. 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).

Consider the following classical network design problem: a set of terminals $T = \{t_i\}$ wishes 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 $\mathsf{NP} \subseteq \mathsf{DTIME}\left(n^{O(\log \log \log n)}\right)$, where $|V| = n$. In particular, we show that this inapproximability threshold holds for (i) the Priority-Steiner Tree problem (ii) the (single-sink) Cost-Distance problem and (iii) the single-sink version of an even more fundamental problem, Fixed Charge Network Flow. 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.

Categories and Subject Descriptors: G.2.2 [**Discrete Mathematics**]: Graph Theory—*Network problems*; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*Computations on discrete structures*

General Terms: Theory, Algorithms

Additional Key Words and Phrases: Hardness of approximation, network design, priority Steiner tree, fixed charge network flow, cost-distance

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 in undirected graphs 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, where n denotes the cardinality of the vertex set:

Fixed-Charge Network Flow (FCNF). The FCNF problem and its many variations have been widely studied in the Operations Research community [Nemhauser and Wolsey 1999; Ortega and Wolsey 2003; Carr et al. 2000; Gunluk 1999]. The *single source* version considered here is a very natural and basic network design problem. The input is an undirected graph $G = (V, E)$ 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 even for the single source version of FCNF.

Priority Steiner Tree. Motivated by the heterogeneity in Quality of Service requirements in multicast (video) applications [Maxemchuk 1997; Turletti and Bolot 1994], the Priority-Steiner Tree problem was defined by Charikar et al. [Charikar et al. 2004]; special cases of this problem have also been studied in the Operations Research community [Current et al. 1986; Duin and Volgenant 1991; Mirchandani 1996; Pirkul et al. 1991].

In the Priority-Steiner Tree problem, given an undirected graph $G = (V, E)$ with edge costs c_e, a set T of terminals and a root node r, we want to find a minimum 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) level $Q(t) \in \{1, 2, \ldots, k\}$ — here 1 is the highest level and k is the lowest level. Furthermore, each edge e offers a certain Quality of Service level $Q(e) \in \{1, 2, \ldots, k\}$, and we want the path in $\mathfrak{T}$ from $t \in T$ to the root r to consist only of edges e with Quality of Service at least as good as $Q(t)$; i.e., $Q(e) \leq Q(t)$.

The best upper bound currently known for the Priority-Steiner Tree problem is a $\min\{2 \ln |T|, 1.55\, k\}$-approximation algorithm [Charikar et al. 2004]; the above paper 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 as a common generalization of several problems in network design and facility location [Meyerson et al. 2000]. 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

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

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

Building on the basic techniques from [Marathe et al. 1998], an $O(\log |T|)$ randomized approximation algorithm for the problem was given in [Meyerson et al. 2000]; this algorithm was subsequently derandomized [Chekuri et al. 2001]. An $O(\log^4 n)$-competitive online algorithm for the problem is also known [Meyerson 2004]. 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 $\Omega(\log \log n)$, where $|V| = n$. In particular, the technical heart of the paper is the following theorem:

Theorem 1.1 *There is no efficient $c \log \log n$ approximation algorithm for the* Priority-Steiner Tree *problem (for some constant c), unless $NP \subseteq$ DTIME $\left(n^{O(\log \log \log n)}\right)$. This holds 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 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 poly-logarithmic inapproximability [Halperin and Krauthgamer 2003] of directed Steiner and group Steiner trees, or of the fact that there were multiple sinks (and hence multiple commodities) in the system [Andrews 2004].

An inapproximability threshold of $\frac{5601}{5600}$ for the basic undirected Steiner tree problem was shown in [Clementi and Trevisan 1999], which immediately implies the same threshold to all the problems we consider. A 1.46 inapproximability threshold for facility location [Guha and Khuller 1999] extends to Cost-Distance, as shown in [Meyerson et al. 2000]. 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 we consider in this paper have $O(\log n)$ approximation algorithms ([Charikar et al. 2004] for Priority-Steiner Tree and [Meyerson et al. 2000] for Cost-Distance). Therefore, our results show the existence of an intermediate class of network design problems which are neither constant-approximable nor poly-logarithmic-inapproximable. At one extreme are the undirected single-source network design problems with homogeneous cost functions on edges such as single-source buy-at-bulk, which admit constant factor approximations [Guha et al. 2000; Talwar 2002; Gupta et al. 2003]. At the other extreme lie the directed Steiner tree problem (whose current inapproximability threshold is $\Omega(\log^{2-\epsilon} n)$ [Halperin and Krauthgamer 2003]), and the multi-commodity buy-at-bulk network design problem (with an inapproximability bound of $\Omega(\log^{\frac{1}{4}-\epsilon} n)$ [Andrews 2004]).

Our problems therefore occupy a middle ground of those problems which admit logarithmic approximations (to date) but have been shown to have 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.

Related Work in Network Design. There has been much research in the area of approximation algorithms for network design, with many new techniques developed and problem areas explored. A partial list includes [Awerbuch and Azar 1997; Andrews and Zhang 2002; Even et al. 2002; Jain 2001; Goemans and Williamson 1997; Salman et al. 2000; Garg et al. 2001; Guha et al. 2000; Guha et al. 2001; Gupta et al. 2003; Gupta et al. 2003; Kumar et al. 2002; Marathe et al. 1998; Melkonian and Tardos 1999; Meyerson et al. 2000; Karger and Minkoff 2000; Ravi and Salman 1999; Ravi and Sinha 2002; Salman et al. 2000; Swamy and Kumar 2002; Talwar 2002]; 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 [Guha et al. 2001; Talwar 2002] and $O(\log n)$-approximable in the multi-commodity case [Awerbuch and Azar 1997; Fakcharoenphol et al. 2003], with an $\Omega(\log^{\frac{1}{4}-\epsilon} n)$-inapproximability threshold known for the multi-commodity case [Andrews 2004]. Constant approximations exist for special cases of multi-commodity concave cost network design, such as the rent-or-buy cost function [Kumar et al. 2002; Gupta et al. 2003]. 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 [Andrews 2004]. 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 inapproximability of these network design problems by using one of them (Priority-Steiner Tree) to encode an instance of the Set Cover problem, which itself is shown to be hard to approximate using a reduction from MAX 3SAT(5). To this end, we begin by explicitly constructing this set cover instance in the next section. In Section 3, we go on to show the hardness of Priority-Steiner Tree using this set system construction. We finally demonstrate the hardness of the other two network design problems in Section 4, using the hardness of Priority-Steiner Tree.

2. CONSTRUCTION OF THE SET SYSTEM

Naturally enough, the hardness reduction for Priority-Steiner Tree involves starting from a 3SAT(5) formula φ and producing an instance of Priority-Steiner Tree. As a crucial building block for our reduction, we use a construction used by Lund and Yannakakis [Lund and Yannakakis 1994] to prove the hardness of the set cover problem. The construction we present in this section is a slight variant of their construction; we give it here both for the sake of completeness, as well as to underscore some properties of the construction that are not explicitly proved in their paper (even though they easily follow from the arguments therein).

The reduction is performed from the gap version of the 3SAT(5) problem (or more precisely, the *Exact MAX 3SAT(5)* problem) which is defined as follows. We are given a CNF formula φ with n variables and $m = 5n/3$ clauses, where each clause contains exactly 3 literals and each variable appears in exactly 5 different clauses. The goal is to find a boolean assignment for the variables which maximizes the number of satisfied clauses. Given some constant ϵ with $0 < \epsilon < 1$, a formula φ is called a *yes-instance* if it is satisfiable, and it is called a *no-instance* if the maximum fraction of clauses that can be satisfied simultaneously is at most $(1-\epsilon)$. Arora et al. showed the following useful corollary of the PCP theorem.

Theorem 2.1 ([Arora et al. 1998]) *There exists a constant $\epsilon \in (0,1)$ such that it is NP-hard to distinguish between the yes-instances and no-instances of 3SAT(5).*

2.1 Parallel Repetition: the Raz Verifier

Starting with such a 3SAT(5) formula φ, we now construct a two-prover proof system with $\ell = \Theta(\log\log\log n)$ repetitions. The verifier (or the *Raz verifier*) in this system receives as an input a 3SAT(5) formula φ, and proceeds as follows.

—As the first step, the verifier chooses a sequence of ℓ clauses $C_1, C_2, \ldots, C_\ell$ independently and uniformly at random from the set of $5n/3$ clauses of φ. It then chooses a sequence of ℓ variables $x_1, \ldots, x_\ell$, where each x_i is chosen independently and uniformly from the three variables participating in C_i; the variable x_i is called the *distinguished variable* of the clause C_i.

—The indices of the clauses $C_1, \ldots, C_{\ell/2}$ and of the variables $x_{\ell/2+1}, \ldots, x_\ell$ are sent to the first prover, while the indices of remaining variables $x_1, \ldots, x_{\ell/2}$ and of the clauses $C_{\ell/2+1}, \ldots, C_\ell$ are sent to the second prover. (Here we are assuming that ℓ is even.)

—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 (for $1 \leq i \leq \ell/2$), the assignment sent by the first prover satisfies the clause, and also that for each clause C_i (for $\ell/2 + 1 \leq i \leq \ell$), the assignment sent by the second prover satisfies the clause. Furthermore, the verifier checks that for each $i \in \{1, 2, \ldots, \ell\}$, the assignments of both provers to the distinguished variable x_i are identical. If any of these tests fail, the verifier rejects, else it accepts if all the tests succeed.

The next theorem follows from the Raz Parallel Repetition Theorem [Raz 1998], and it shows that asking each of the two provers ℓ questions in one round (instead of having ℓ rounds with a single independent random question in each round) still causes the verifier's error probability to go down exponentially in ℓ.

Theorem 2.2 *If φ is a yes-instance of 3SAT(5), then there is a strategy of the two provers that makes the verifier always accept. Else if φ is a no-instance, then for any strategy of the provers, the acceptance probability of the verifier in the above scheme is at most $2^{-\alpha\ell}$, for some universal constant α.*

Let X and Y be the sets of all possible queries to provers 1 and 2 respectively; since each query is an ordered sequence, these sets are disjoint—and furthermore, given a query q, one can infer whether q belongs to X or to Y. Note that each such query contains an $\ell/2$-tuple of clauses and an $\ell/2$-tuple of variables, and hence $|X| = |Y| \leq (5n/3)^{\ell/2} \cdot n^{\ell/2} < (2n)^\ell$.

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; clearly, $|A(q)| \leq 7^{\ell/2} \cdot 2^{\ell/2} < 8^\ell$.

We denote by R the set of all the random strings of the verifier; thus $|R| \leq (5n)^\ell$. For a random string $r \in R$, let $q_1(r) \in X$ and $q_2(r) \in Y$ be the queries sent to the two provers when the verifier chooses r. Clearly, r defines a *constraint* between the answers to $q_1(r)$ and $q_2(r)$. In what follows, we refer to r as both a random string and a *constraint*, interchangeably.

2.2 The Set System

We are now in a position to specify the set system $\mathcal{S}$ corresponding to the 3SAT(5) formula φ. We will define a universe of elements U, and the set system $\mathcal{S} \subseteq 2^U$ will consist of subsets of U.

—**The Sets.** The set system $\mathcal{S}$ contains a set $S(q, a)$ for each query $q \in X \cup Y$ and for each answer $a \in A(q)$; i.e.,

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

Hence the number of sets is bounded by $2 \cdot (2n)^\ell \cdot 8^\ell < (32n)^\ell$.

—**The Elements.** Consider a random string $r \in R$, and let $x_1, \ldots, x_\ell$ be the sequence of distinguished variables that correspond to r. Let $\mathcal{A}$ be the set of all the possible assignments to these variables, and thus $|\mathcal{A}| = 2^\ell$. For each $\mathcal{A}' \subseteq \mathcal{A}$, we define a new element $E(r, \mathcal{A}')$; the universe U consists of all these elements. Since $|R| \leq (5n)^\ell$, and since for each $r \in R$ there are 2^{2^ℓ} elements, the total number of elements in U is bounded by $(5n)^\ell \cdot 2^{2^\ell}$.

—**Element-Set Inclusions.** An element $E(r, \mathcal{A}') \in U$ is defined to belong to all the sets $S(q_1(r), a) \in \mathcal{S}$ such that the answer a is consistent with some assignment in $\mathcal{A}'$; in other words, the projection of the bits in a onto the distinguished variables belongs to the set $\mathcal{A}'$.
Furthermore, $E(r, \mathcal{A}')$ belongs to all the sets $S(q_2(r), a') \in \mathcal{S}$ such that a' is consistent with some assignment in $\overline{\mathcal{A}'}$; i.e., the projection of a' onto the distinguished variables **does not** lie in $\mathcal{A}'$.

Now that we have described the set system, let us delineate some of the important properties that will be used in the rest of the reduction.

Lemma 2.3 (Set Sizes) *The size of any set in $\mathcal{S}$ is $z = \frac{1}{2} \cdot 15^{\ell/2} \cdot 2^{2^\ell}$.*

PROOF. Let us figure out the size of any set $S(q, a)$ for some $q \in X$. (The size for sets with $q \in Y$ will be the same.) Observe that for any query $q \in X$, there are $15^{\ell/2}$ random strings $r \in R$ such that $q = q_1(r)$; indeed, given the query q, one can infer the identity of the string r by guessing the distinguished variables for the clauses in q (for which there are $3^{\ell/2}$ choices), and the clauses to which the variables in q belong (for which there are $5^{\ell/2}$ choices).

Now, there are 2^{2^ℓ} elements $E(r, A')$ that correspond to any fixed random string r, and given an assignment $a \in A(q)$, exactly half of these elements belong to set $S(q, a)$. Hence, the size of any set $S(q, a) \in \mathcal{S}$ is equal to $z = \frac{1}{2} \cdot 15^{\ell/2} \cdot 2^{2^\ell}$. □

Lemma 2.4 (Element Degrees) *Each element $e = E(r, \mathcal{A}') \in U$ belongs to $d_e \leq 2^{2\ell}$ sets in $\mathcal{S}$.*

PROOF. Given an element $e = E(r, \mathcal{A}')$, let $\mathcal{A}$ be the set of all the assignments to the distinguished variables corresponding to random string r (and hence $\mathcal{A}' \subseteq \mathcal{A}$). Given an assignment $\hat{a} \in \mathcal{A}$ to the distinguished variables, let $\mathcal{A}_1(\hat{a})$ and $\mathcal{A}_2(\hat{a})$ be the subsets of answers to queries $q_1(r)$ and $q_2(r)$, respectively, which are consistent with $\hat{a}$. Notice that $|\mathcal{A}_1(\hat{a})|, |\mathcal{A}_2(\hat{a})| \leq 2^\ell$: there are exactly $4^{\ell/2} = 2^\ell$ answers a of

Prover 1 to query $q_1(r)$, and of Prover 2 to query $q_2(r)$, that are consistent with assignment $\hat{a}$—though some of them may not satisfy all the query clauses. Indeed, the projection of a on the distinguished variables must correspond to $\hat{a}$, which leaves us with $2 \cdot (\ell/2) = \ell$ variables that appear in query clauses of $q_1(r)$, or of $q_2(r)$, that are not distinguished.

If $\hat{a} \in \mathcal{A}'$ then $e = E(r, \mathcal{A}')$ belongs to all the sets $S(q_1(r), a)$ for all $a \in \mathcal{A}_1(\hat{a})$, and if $\hat{a} \in \overline{\mathcal{A}'}$, then e belongs to all the sets $S(q_2(r), a)$, where $a \in \mathcal{A}_2(\hat{a})$. Furthermore, these are the only sets in $\mathcal{S}$ that contain e. As $|\mathcal{A}| = 2^\ell$, we have that $d_e \leq 2^\ell \cdot |\mathcal{A}| = 2^{2\ell}$. □

Let the *degree* d_e of an element in $e \in U$ be the number of sets in $\mathcal{S}$ that contain e. If we let $d = \max_e\{ d_e\}$ denote the maximum degree of an element in U, then $d \leq 2^{2\ell}$ by Lemma 2.4, and thus number of elements is $|U| \geq (|\mathcal{S}|z)/d$.

2.3 The Set Cover Gap

We want to show that if φ is a yes-instance, then there is a set cover for U of size at most $|X| + |Y|$, whereas if φ is a no-instance, then even "large" covers must leave many elements in U uncovered.

Lemma 2.5 *If φ is a yes-instance, then there exists a set cover containing at most $|X| + |Y|$ sets.*

Proof. Since φ is a yes-instance, there is a strategy of the provers which makes the verifier always accept. For instance, they can choose some satisfying assignment a_{sat} for φ in a consistent matter (say the lexicographically first satisfying assignment), and then answer queries according to the settings in a_{sat}.

We now use this strategy to choose the set cover: for each query $q \in X \cup Y$, choose the set $S(q, a)$, where a is the answer to query q under this strategy. To see that this is a cover, consider any element $E(r, \mathcal{A}')$: let $\hat{a}$ be the projection of a_{sat} on the ℓ distinguished variables that correspond to this random string r, let a_1 be the projection of a_{sat} on the variables in $q_1(r)$, and let a_2 be its projection on variables in $q_2(r)$. By its very construction, $\hat{a}$ is consistent with both a_1 and a_2.

Finally, note that both $S(q_1(r), a_1)$ and $S(q_2(r), a_2)$ belong to the set cover we have picked. Furthermore, if $\hat{a} \in \mathcal{A}'$, then the set $S(q_1(r), a_1)$ covers this element, Else if $\hat{a} \notin \mathcal{A}'$, set $S(q_2(r), a_2)$ covers this element. □

The proof of the other case is more involved: if we assume that φ is a no-instance, we need to show that even if we choose a "large" number of sets, there is still a substantial fraction of elements not covered by these sets.

Theorem 2.6 *Consider any sub-family $\mathcal{S}' \subseteq \mathcal{S}$ containing less than $(|X| + |Y|) \cdot h$ sets for some integer $h > 1$, where $64\, h^2 < 2^{\alpha\ell}$. Then, at least a fraction $\frac{1}{4 \cdot 2^{8h}}$ of the elements in U are not covered by sets in $\mathcal{S}'$.*

Proof. Let $Q \subseteq X \cup Y$ be the subset of queries q, such that for each $q \in Q$, the number of sets $S(q, a) \in \mathcal{S}'$ is at least $4h$. Since the size of $\mathcal{S}'$ is at most h times $|X \cup Y|$, a simple averaging argument shows that Q contains at most a quarter of $X \cup Y$.

Let $R' \subset R$ be the subset of random strings for which both the queries $q_1(r) \notin Q$ and $q_2(r) \notin Q$. Since each query participates in the same number of constraints, $|R'| \geq \frac{1}{2}|R|$. In other words, each query in $X \cup Y$ has the same probability of being generated, and hence the chance that we hit $Q \subseteq X \cup Y$ is at most $\Pr[q_1(r) \in Q] + \Pr[q_2(r) \in Q] \leq 2|Q|/|X \cup Y| \leq 1/2$.

Let $R'' \subseteq R'$ be a further subset of these random strings such that there exist 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 claim.

Claim 2.7 $|R''| \leq \frac{1}{2}|R'|$.

Proof of Claim 2.7. To prove this, we assume the converse, and use this to show a strategy that makes the verifier accept the no-instance φ with high probability; this gives the desired contradiction. Indeed, consider the following strategy: on receiving a query $q \in (X \cup Y) \setminus Q$, the prover in question chooses randomly one of its assignments a (at most $4h$) that correspond to the sets in $S(q, a) \in \mathcal{S}'$.

Since we assumed that $|R''| \geq \frac{1}{2}|R'| \geq \frac{1}{4}R$, the probability that the verifier chooses a random string in R'' is at least $\frac{1}{4}$; moreover, if a string in R'' is chosen, the probability that the answers of the provers are consistent is at least $1/(16h^2)$. Hence, the verifier accepts with probability at least $1/(64h^2) > 2^{-\alpha l}$, contradicting Theorem 2.2. □

Let us now consider some random string $r \in R' \setminus R''$: recall that since $r \in R'$, the cover $\mathcal{S}'$ contains fewer than $4h$ sets from the family $\{S(q_1(r), a) \mid a \in A(q_1(r))\}$, as well as fewer than $4h$ sets from the family $\{S(q_2(r), a') \mid a' \in A(q_2(r))\}$; furthermore, since $r \notin R''$, there is no pair of sets $S(q_1(r), a)$ and $S(q_2(r), a')$ in $\mathcal{S}'$ where a and a' are consistent.

Consider the distinguished variables corresponding to r, and let A'_1 be the subset of assignments to these distinguished variables so that for each $a \in A'_1$, there is some set $S(q_1(r), a') \in \mathcal{S}'$ where a' is consistent with a. Define A'_2 similarly. Note that the properties of R'' ensure that both $|A'_1|, |A'_2| \leq 4h$, and furthermore that $A'_1 \cap A'_2 = \emptyset$.

Let $e = E(r, \mathcal{A}')$ be some element in U: it is covered by some set in $\mathcal{S}'$ if and only if either $\mathcal{A}' \cap A'_1 \neq \emptyset$, or $\overline{\mathcal{A}'} \cap A'_2 \neq \emptyset$. Now, 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. Hence, if we consider the assignments in A'_1 and A'_2 (at most $8h$), the fraction of elements corresponding to random string r, and are not covered, is at least 2^{-8h}. Finally, since the size of $R' \setminus R''$ is at least $\frac{1}{4}|R|$ (by Claim 2.7), the total fraction of elements not covered by $\mathcal{S}'$ is at least $(1/4) \cdot 2^{-8h}$, completing the proof of Theorem 2.6.

While one can use the construction given above to prove hardness results for the Set Cover problem, recall that our goal is to use this to prove hardness results for the Priority-Steiner Tree problem, which is the subject of the next section.

3. HARDNESS OF THE PRIORITY STEINER TREE PROBLEM

To begin, let us recall the Priority-Steiner Tree problem. In this, we are given an undirected graph $G = (V, E)$ with edge costs (or lengths) $c_e \in \mathbb{R}_{\geq 0}$ and edge prior-

ities $Q(e) \in \{1, 2, \ldots, k\}$. We 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. We are also given 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 Steiner tree $\mathfrak{T}$ of minimum cost that spans the root r and all the terminals in T, and where each edge e on the unique path in $\mathfrak{T}$ from each $t \in T$ to r has a priority $Q(e) \leq Q(t)$.

The hardness result will proceed along standard lines: given a 3SAT(5) instance φ, we want to define a graph which is an instance of the Priority-Steiner Tree problem; to this end, we will use the set system $\mathcal{S}$ over the universe U which we constructed previously in Section 2.2.

3.1 The Reduction from φ: Constructing the Graph

The graph G_φ consists of a root vertex r and a collection of $|\mathcal{S}|$ disjoint sets of *non-terminal* vertices $V_1, V_2, \ldots, V_{|\mathcal{S}|}$, with each set having cardinality $M + 1$; here M is a large integer whose value will be determined later. The vertices in each such set V_i will be denoted by $v^i_0, \ldots, v^i_M$. For each i in the set $\{1, 2, \ldots, |\mathcal{S}|\}$ and for each pair of successive vertices (v^i_j, v^i_{j+1}) (with $0 \leq j < M$), there are k parallel edges connecting v^i_j and v^i_{j+1}, each with a different priority level from 1 to k. The costs (or lengths) of all these edges is set to $\frac{1}{|\mathcal{S}|M}$.

In other words, for each $i : 1 \leq i \leq |\mathcal{S}|$, there is a path of length $\frac{1}{|\mathcal{S}|}$ representing the set $S_i \in \mathcal{S}$. This path is denoted P_i and it consists of many (in particular, M) small edges. Since the cost (or length) of each edge in path P_i is $\frac{1}{|\mathcal{S}|M}$, the total length of all these edges is 1. Finally, for each path P_i, each edge belonging to P_i is replaced by k parallel edges, each having a different priority level.

Apart from these edges and non-terminal vertices described above, there is a set T_j of level-j terminal vertices and a set E_j of level-j edges for each priority level $j \leq k$; the cost of each edge in E_j will be 0. The terminal sets T_j and edge sets E_j are recursively defined using the set system $\mathcal{S}$, starting from the highest priority level 1.

Level-1 Terminals: For every element $e \in U$ in the set system, we define a level 1 terminal $t^1(e)$ belonging to T_1.

Level-1 Edges: For $1 \leq i \leq |\mathcal{S}|$, consider the path P_i corresponding to set $S_i \in \mathcal{S}$. Recall that $|S_i| = z$, and let $S_i = \{e_{i_1}, e_{i_2}, \ldots, e_{i_z}\}$. Let us subdivide P_i into $2z$ subpaths of equal length, and denote the endpoints of these subpaths by $p_0, p_1, \ldots, p_{2z}$. (See Figure 1 for an illustration.) Connect each one of the vertices $p_2, p_4, \ldots, p_{2z-2}$ to the root; moreover, for each value of $a : 1 \leq a \leq z$, connect the vertex p_{2a-1} to the terminal $t^1(e_{i_a})$. All these newly added edges belong to E_1, and hence have priority level 1; recall that all these edges have cost (length) 0.

Note that each path P_i has been divided into $2z$ subpaths: these subpaths will be called the *level-1 subpaths of* P_i. Each one of these paths will, in turn, be divided into $2z$ equal-length subpaths, which will be called the *level-2 subpaths of* P_i. Continuing this recursively, the path P_i is divided into $n_j = (2z)^j$ *level-j subpaths* for each priority level $j \leq k$. Let us now define the construction for level-j

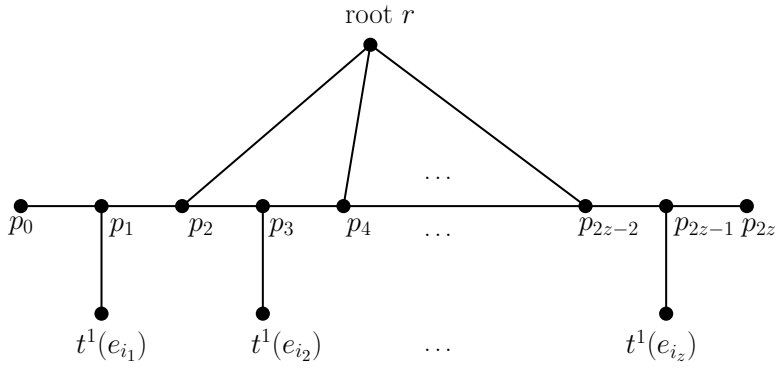

Fig. 1. Level 1 construction for path P_i.

edges E_j and terminal sets T_j.

Level-j Terminals: Recall that each path P_i was divided into $n_{j-1} = (2z)^{j-1}$ level-$(j-1)$ subpaths. For the level-j construction, we use n_{j-1} copies of the set system $\mathcal{S}$: for each $i \in [|\mathcal{S}|]$ and $b \in [n_{j-1}]$, the b^{th} level-$(j-1)$ subpath of P_i will represent the set S_i in the b^{th} set system.
To define the set T_j of level-j terminals, we create a terminal $t_b^j(e)$ for each element e in the b^{th} set system for $1 \leq b \leq n_{j-1}$. Of course, each of these terminals is at priority level j, and there are $n_{j-1}|U|$ terminals in T_j.

Level-j Edges: Given some $i \in [|\mathcal{S}|]$ and $b \in [n_{j-1}]$, consider the b^{th} level-$(j-1)$ subpath of P_i. Much as in the level-1 case, let us divide this subpath into $2z$ level-j subpaths of equal length, and 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 the vertex p_{2a-1} to the terminal $t_b^j(e_{i_a})$. The newly added edges all have priority level-j and zero cost.

This completes the reduction of the 3SAT(5) instance φ (via the set system $\mathcal{S}$) to the instance G_φ of Priority-Steiner Tree. The following sections will give the analysis of this reduction, and also indicate how to set the parameters including the number of levels k, the length M of the paths, and the parameter ℓ of the set-system $\mathcal{S}$.

3.2 Analyzing the Gap

Before we begin the formal analysis of the above construction, it bears repeating that the only edges with non-zero cost are the edges that lie within the paths $P_1, \ldots, P_{|\mathcal{S}|}$. Furthermore, all these edges have the same cost $1/(M|\mathcal{S}|)$, including the parallel edges which have different priority levels, and thus an optimal solution will obviously buy only level-1 edges on these paths. However, any such solution will also use some zero-cost edges in $E_1, \ldots, E_k$ of priority levels $1, \ldots, k$, and it is the structure of these edge sets that we intend to exploit.

Indeed, the basic idea underlying the construction is the following. For any yes-instance φ, the optimal solution to the instance G_φ can just buy a small fraction of the edges on paths $P_1, P_2, \ldots, P_{|\mathcal{S}|}$: for each set S_i in the optimal set cover solution to $\mathcal{S}$, it can just buy all the level-1 edges on path P_i. However, in the case of a

no-instance, any solution for Priority-Steiner Tree on G_φ will end up buying a much larger fraction of edges on the paths $P_1, \ldots, P_{|\mathcal{S}|}$, giving us the desired gap.

3.2.1 *Analysis for Yes-instances φ.* Let us first show the easy half of the result: there is a low-cost solution for instances corresponding to yes-instances.

Theorem 3.1 (Cost for yes-instances) *If φ is a yes-instance, then there is a solution of the corresponding* Priority-Steiner Tree *problem instance G_φ with cost at most $\frac{|X|+|Y|}{|\mathcal{S}|}$.*

Proof. Since φ is a yes-instance, Lemma 2.5 guarantees a sub-family $\mathcal{S}' \subset \mathcal{S}$ with $|\mathcal{S}'| = |X| + |Y|$ that covers all the elements of U. Now we can define a solution to the Priority-Steiner Tree instance G_φ: for each $S_i \in \mathcal{S}'$, we take all the priority level-1 edges on the path P_i, and also all the edges in $E_1, E_2, \ldots, E_k$ that are adjacent to some vertex on path P_i.

To see that this solution is a feasible one, consider a priority level $j \in [k]$ and any value $b \in [n_{j-1}]$: since the family $\mathcal{S}'$ is a feasible set cover, all the elements of the b^{th}-set system are connected to the root r by the chosen edges. Finally, since we buy $|X| + |Y|$ paths P_i, each of length $1/|\mathcal{S}|$, the cost of the solution is bounded by $\frac{|X|+|Y|}{|\mathcal{S}|}$. □

3.2.2 *Analysis for No-instances φ.* For the case of no-instances, we will prove the following theorem:

Theorem 3.2 (Cost for no-instances) *Given a no-instance φ, if the number of priority levels k in the above construction is $16 \cdot 2^{8h} \cdot 2^{2\ell}$, then the cost of any solution is at least $\frac{h}{2|\mathcal{S}|}(|X| + |Y|)$.*

The rest of Section 3.2.2 will be devoted to proving Theorem 3.2. But before we dive into the proof, note that there is a gap of $h/2$ in the costs promised by Theorems 3.1 and 3.2, and if we set the parameters $h = \Theta(\log \log n)$ and $\ell = \Theta(\log \log \log n)$, we get a $\Omega(\log \log n)$ hardness result. (The details appear in Section 3.2.3.)

Let us fix a solution $\mathfrak{T}$ of minimum cost to the instance G_φ: it is easy to check that any solution of minimum cost contains no cycles. To simplify the analysis further, we make the following assumptions on the structure of $\mathfrak{T}$; these assumptions can be made without loss of generality.

—The solution $\mathfrak{T}$ uses the edges with the lowest possible priority levels. In other words, if the solution contains edge e of priority level j, then using the priority level $(j+1)$ edge parallel to e, instead of the edge e itself, makes the solution infeasible.

—Each level-j terminal $t \in T_j$ is connected to the root r via a level-j subpath of one of the paths $P_1, P_2, \ldots, P_{|\mathcal{S}|}$; recall that each such level-j subpath has length $1/(n_j \times |\mathcal{S}|) = 1/(|\mathcal{S}|(2z)^j)$. Furthermore, different terminals of the same priority level are connected via disjoint subpaths: for example, the path from terminal $t_b^j(e_{i_a})$ goes from the terminal via p_{2a-1} and then p_{2a-2} or p_{2a} to the root, where the segments (p_{2a-1}, p_{2a}) and (p_{2a-2}, p_{2a-1}) are level-j subpaths.

Therefore, given a path P_i with $i \in [|\mathcal{S}|]$, for each level-j subpath p of P_i (with $j \in [k]$) and for each level $j' \leq j$, either all the edges of level-j' on subpath p are in the solution, or none of them is.

For any $j \in [k]$, let c_j denote the cost of all the edges in the solution $\mathfrak{T}$ whose priority levels lie in $\{1, 2, \ldots, j\}$. The following lemma captures the crucial properties of the reduction.

Lemma 3.3 *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. If the cost $c_{j-1} < \frac{h}{2|\mathcal{S}|}(|X| + |Y|)$ for some priority level $j : 1 < j \leq k$, 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 a path P_i for some $i \in [|\mathcal{S}|]$, and let p be the b^{th} level-$(j-1)$ subpath of P_i for some $b \in [n_{j-1}]$. Recall that this subpath p represents set S_i in the b^{th} set system of priority level j: we say that this set is *pre-chosen* if and only if all the edges of p of priority level j' are in the Priority-Steiner Tree solution for some priority level $j' < j$. Note that by our assumptions above, if this set is not pre-chosen, the Priority-Steiner Tree solution contains no edge of levels in $\{1, \ldots, j-1\}$ on the path p.

Since the cost $c_{j-1} < \frac{h}{2|\mathcal{S}|}(|X| + |Y|)$, and the cost of the path representing each such set is $\frac{1}{|\mathcal{S}|\,(2z)^{j-1}} = \frac{1}{|\mathcal{S}|\,n_{j-1}}$, the total number of level-j sets that are pre-chosen by the solution is at most $\frac{h}{2}(|X| + |Y|) \cdot n_{j-1}$. Hence, in at least half of the n_{j-1} level-j set cover instances, fewer than $h(|X| + |Y|)$ sets are pre-chosen. In each one of these instances, Theorem 2.6 implies that a fraction of at least $\frac{1}{4 \cdot 2^{8h}}$ of the elements do not belong to the sets pre-chosen by the solution. Now 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; moreover, 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 lemma. □

We can now use this lemma to prove that any solution for G_φ has cost at least $\frac{h}{2} \cdot \frac{|X|+|Y|}{|\mathcal{S}|}$ when φ is a no-instance, as claimed in Theorem 3.2.

Proof of Theorem 3.2. Suppose a solution for G_φ has cost less than $C = \frac{h}{2} \cdot \frac{|X|+|Y|}{|\mathcal{S}|}$. Then, for each priority level $j : 1 < j \leq k$, the cost $c_{j-1} < C$, and hence by Lemma 3.3 the cost of the priority level-j edges in the solution is at least $1/(16 \cdot 2^{8h} \cdot 2^{2\ell}) = 1/k$.

But, we have $k = 16 \cdot 2^{8h} \cdot 2^{2\ell}$ priority levels, and if the edges at each of these levels cost at least $1/k$, the total cost is $1 \gg C$, which is a contradiction. Hence,

the total cost of any solution for G_φ must be at least $C = \frac{h}{2|\mathcal{S}|}(|X| + |Y|)$, proving the theorem. □

3.2.3 *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 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 set $h = \Theta(\log\log n)$, 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 a factor of $\frac{h}{2} = \Theta(\log\log N)$, we have proved the following theorem:

Theorem 3.4 *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 the priority Steiner tree problem where all 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 the priority Steiner problem 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 undirected graph G with a specified root vertex r. Each vertex 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 $\sum_e x_e$, 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 $u_e = n^{5(k-i)+2}$.

[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 by a multi-edge.

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 set $x_e = 0$. 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 $U(e)$ is 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 the vertices 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 than 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 vertices and each terminal vertex 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 $\mathfrak{T}_P$ is infeasible. That is, for some terminal t of priority i, there is a cut that separates it from the root such that 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 $n^{5(k-i)}$, whereas the capacities of these edges are at most $n^{5(k-i)-3}$. 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 3.4.

Theorem 4.1 *The Fixed-Charge Network Flow problem is $\Omega(\log \log n)$-hard to approximate, even in the single-source 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 telecommunications 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 an undirected graph rooted at r, as follows. Each edge e has a length l_e and a cost c_e, and each vertex v has demand d_v. A feasible solution consists of a tree $\mathfrak{T}_C$ spanning all vertices with positive demand and the root. Let $V(e)$ be the set of vertices whose paths to the root use edge e. The total cost of the solution is given by $c(\mathfrak{T}_C) = \sum_e (c_e + l_e \sum_{v \in V(e)} d_v)$. 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 Cost-Distance by defining the length l_e of edge e to be $l_e = c_e/u_e$. The costs of the edges and the demands at the vertices are unchanged.

Theorem 4.2 *The* Cost-Distance *problem cannot be approximated 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 [Meyerson et al. 2000; Garg et al. 2001] (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 is, 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. □

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 [Guha et al. 2001; Talwar 2002]. 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, who proved stronger inapproximability results for non-uniform multi-commodity buy-at-bulk network design [Andrews 2004].

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, there is a gap between the approximability of Cost-Distance and Priority-Steiner Tree (both admit $O(\log n)$-approximation algorithms) and their inapproximability (which is $\Omega(\log\log n)$ due to the results of this paper). In fact, using instances similar to those used here, one can show that the integrality gap of the natural linear-programming relaxation for the Priority-Steiner Tree problem is $\Omega(\frac{\log n}{\log\log n})$; can we obtain stronger linear-programming relaxations for these problems? Furthermore, 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 wide open. Finally, designing networks on *directed* graphs presents several challenges which are as yet poorly understood.

ACKNOWLEDGMENTS

We would like to thank Moses Charikar, Chandra Chekuri, Kedar Dhamdhere, Jochen Könemann, Amit Kumar, and especially Bruce Shepherd for many useful conversations.

REFERENCES

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

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

Arora, S., Lund, C., Motwani, R., Sudan, M., and Szegedy, M. 1998. Proof verification and the hardness of approximation problems. *J. ACM 45,* 3, 501–555.

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

Carr, R. D., Fleischer, L., Leung, V. J., and Phillips, C. A. 2000. Strengthening integrality gaps for capacitated network design and covering problems. In *Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete*. 106–115.

Charikar, M., Naor, J. S., and Schieber, B. 2004. Resource optimization in QoS multicast routing of real-time multimedia. *IEEE/ACM Transactions on Networking 12,* 2, 340–348.

Chekuri, C., Khanna, S., and Naor, J. S. 2001. A deterministic algorithm for the cost-distance problem. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms*. 232–233.

Clementi, A. E. and Trevisan, L. 1999. Improved non-approximability results for minimum vertex cover with density constraints. *Theoretical Computer Science 225,* 1–2, 113–128.

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

Duin, C. and Volgenant, T. 1991. The multi-weighted Steiner tree problem. *Ann. Oper. Res. 33,* 1-4, 451–469. Topological network design (Copenhagen, 1989).

Even, G., Kortsarz, G., and Slany, W. 2002. On network design: fixed charge flows and the covering steiner problem. In *Proceedings of the 8th Scandinavian Workshop on Algorithm Theory*. Lecture Notes in Computer Science, vol. 2368. Springer, 318–329.

Fakcharoenphol, J., Rao, S., and Talwar, K. 2003. A tight bound on approximating arbitrary metrics by tree metrics. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*. 448–455.

Garg, N., Khandekar, R., Konjevod, G., Ravi, R., Salman, F. S., and Sinha, A. 2001. 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*. Lecture Notes in Computer Science, vol. 2081. 170–184.

Goemans, M. X. and Williamson, D. P. 1997. The primal-dual method for approximation algorithms and its application to network design problems. In *Approximation Algorithms for NP-hard Problems*, D. S. Hochbaum, Ed. PWS Publishing.

Guha, S. and Khuller, S. 1999. Greedy strikes back: Improved facility location algorithms. *Journal of Algorithms 31,* 1, 228–248.

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

Guha, S., Meyerson, A., and Mungala, K. 2001. 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)*. 383–388.

Gunluk, O. 1999. A branch-and-cut algorithm for capacitated network design problems. *Mathematical Programming 86*, 17–39.

Gupta, A., Kumar, A., Pál, M., and Roughgarden, T. 2003. Approximations via cost-sharing. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*. 606–615.

Gupta, A., Kumar, A., and Roughgarden, T. 2003. Simpler and better approximation algorithms for network design. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*. 365–372.

Halperin, E. and Krauthgamer, R. 2003. Polylogarithmic inapproximability. In *Proceedings of the* 35*th Annual ACM Symposium on Theory of Computing*. 585–594.

Jain, K. 2001. A factor 2 approximation algorithm for the generalized Steiner network problem. *Combinatorica 21,* 1, 39–60.

Karger, D. R. and Minkoff, M. 2000. Building Steiner trees with incomplete global knowledge. In *Proceedings of the 41th Annual IEEE Symposium on Foundations of Computer Science*. 613–623.

Kumar, A., Gupta, A., and Roughgarden, T. 2002. 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*. 333–342.

Lund, C. and Yannakakis, M. 1994. On the hardness of approximating minimization problems. *J.ACM 41,* 5, 960–981.

Marathe, M. V., Ravi, R., Sundaram, R., Ravi, S. S., Rosenkrantz, D. J., and Hunt, III, H. B. 1998. Bicriteria network design problems. *J. Algorithms 28,* 1, 142–171.

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

Melkonian, V. and Tardos, É. 1999. Approximation algorithms for a directed network design problem. In *Integer programming and combinatorial optimization (Graz, 1999)*. Lecture Notes in Comput. Sci., vol. 1610. Springer, Berlin, 345–360.

Meyerson, A. 2004. Online algorithms for network design. In *Proceedings of the 16th ACM Symposium on Parallelism in Algorithms and Architectures*. 275–280.

Meyerson, A., Munagala, K., and Plotkin, S. 2000. Cost-distance: Two metric network design. In *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science*. 624–630.

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

Nemhauser, G. L. and Wolsey, L. A. 1999. *Integer and Combinatorial Optimization*.

Ortega, F. and Wolsey, L. A. 2003. A branch-and-cut algorithm for the single-commodity, uncapacitated, fixed-charge network flow problem. *Networks 41,* 3, 143–158.

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

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

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

Raz, R. 1998. A parallel repetition theorem. *SIAM J. Comput. 27,* 3, 763–803.

Salman, F. S., Cheriyan, J., Ravi, R., and Subramanian, S. 2000. Approximating the single-sink link-installation problem in network design. *SIAM Journal on Optimization 11,* 3, 595–610.

Swamy, C. and Kumar, A. 2002. Primal-dual algorithms for the connected facility location problem. In *Proceedings of the 5th International Workshop on Approximation Algorithms for*

Combinatorial Optimization Problems (APPROX). Lecture Notes in Computer Science, vol. 2462. 256–269.

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

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

Received Month Year; revised Month Year; accepted Month Year