Improved approximation algorithms for network design problems*

M.X. Goemans† A.V. Goldberg‡ S.A. Plotkin§ D.B. Shmoys¶ É. Tardos‖
D.P. Williamson**

Abstract

We consider a class of network design problems in which one needs to find a minimum-cost network satisfying certain connectivity requirements. For example, in the survivable network design problem, the requirements specify that there should be at least $r(v,w)$ edge-disjoint paths between each pair of vertices v and w. We present an approximation algorithm with a performance guarantee of $2\mathcal{H}(f_{\max}) = 2(1+\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{f_{\max}})$ where $f_{\max}$ is the maximum requirement. This improves upon the best previously known performance guarantee of $2f_{\max}$. We also show that our analysis is tight, up to a constant factor. In addition, we present approximation algorithms for two natural variations of the problem involving capacities. Our algorithms are based on a primal-dual approximation method for approximating a class of integer linear programs.

1 Introduction

Network design problems have a wide range of practical applications, ranging from telecommunications to transshipment problems [10]. In this paper, we will investigate approximation algorithms for several basic network design problems. In a network design problem, the input consists of an undirected graph $G = (V,E)$, where each edge $e \in E$ has a nonnegative cost $c(e)$, and we wish to select a minimum-cost subgraph that satisfies certain specified connectivity requirements. A general way to specify these constraints is to require, for each $S \subseteq V$, that the subgraph must contain at least $f(S)$ edges in the cut $\delta(S) = \{(v,w) \in E : v \in S, w \notin S\}$, where we assume that there is a polynomial-time subroutine to compute f.

*A preliminary version of this paper appeared in the Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms, Arlington, VA, pages 223–232m 1994.

†Department of Mathematics, Massachusetts Institute of Technology; research supported in part by Air Force contract F49620-92-J-0125 and DARPA contract N00014-92-J-1799.

‡Department of Computer Science, Stanford University; research supported in part by ONR Young Investigator Award N00014-91-J-1855, NSF Presidential Young Investigator Grant CCR-8858097 with matching funds from AT&T, DEC, and 3M, and a grant from Powell Foundation.

§Department of Computer Science, Stanford University; research supported by U.S. Army Research Office Grant DAAL-03-91-G-0102 and by a grant from Mitsubishi Electric Laboratories.

¶School of Operations Research and Industrial Engineering, Cornell University; research partially supported by NSF PYI Award CCR-89-96272 with matching support from UPS, Sun, Proctor & Gamble, and DuPont. This research was done during an extended visit to DIMACS.

‖School of Operations Research and Industrial Engineering, Cornell University; research supported in part by a Packard Fellowship, an NSF PYI award, and by the National Science Foundation, the Air Force Office of Scientific Research, and the Office of Naval Research, through NSF grant DMS-8920550. This research was done during an extended visit to Princeton University.

**IBM T.J. Watson Research Center. This research was done while a graduate student at MIT and supported by an NSF Graduate Fellowship, Air Force contract F49620-92-J-0125, and DARPA contract N00014-92-J-1799.

Type of Function f	Guarantee with Multiple Copies Allowed	Guarantee with Multiple Copies Forbidden
$r(u,v) = \min\{r(u), r(v)\}$	$\min\{2q, 2\mathcal{H}(f_{\max})\}$ [6]	–
$r(u,v)$	$2\log_2 f_{\max}$ [2]	–
{0,2}-proper	–	3 [9]
proper	$2\mathcal{H}(f_{\max})$ [7] $2\log_2 d$ [1]	$2f_{\max}$ [14] $2\mathcal{H}(f_{\max})$ this paper

Figure 1: Summary of the performance guarantees for uncapacitated network design algorithms

There has been a great deal of recent attention on the design and analysis of approximation algorithms for network design problems. There are two primary variants of these problems: one in which a solution may include multiple copies of each edge, and one in which this is forbidden. We first focus on the latter case. Extending work of Klein & Ravi for proper function with range $\{0,2\}$, Williamson, Goemans, Mihail, & Vazirani [14] gave a polynomial-time $2f_{\max}$-approximation algorithm when the function f is proper, where $f_{\max} = \max_{S \subseteq V} f(S)$ and a *ρ-approximation algorithm* is an algorithm that always delivers a solution of cost at most ρ times the optimum. We shall defer the definition of proper functions, but we note that this formulation includes a broad range of applications, such as the minimum Steiner tree problem, the minimum k-edge-connected subgraph problem, and the minimum T-join problem. Our main result is to give a polynomial-time $2\mathcal{H}(f_{\max})$-approximation algorithm for this problem, where $\mathcal{H}(f_{\max}) = 1 + 1/2 + 1/3 + \cdots + 1/f_{\max}$ is the harmonic function; this represents an exponential improvement in the performance guarantee. The analysis of the performance guarantee is tight, and the examples to prove this suggest that a rather different approach is needed to obtain an algorithm with a constant performance guarantee.

In each of the recent results on approximation algorithms for the variant of the network design problem in which multiple copies of each edge are allowed, the core of the result lies in the design and analysis of an algorithm for the special case in which the function f has range $\{0,1\}$. (Note that for $\{0,1\}$-functions, there is no longer any difference between the variant in which multiple copies of edges are allowed and the variant in which they are forbidden.) In the *survivable network design problem*, the connectivity requirements are specified as follows: for each pair of nodes $v, w \in V$, there must be $r(v,w)$ edge-disjoint paths that connect v and w. This is the special case of our network design problem in which $f(S) = \max_{v \in S, w \notin S} r(v,w)$. Goemans & Bertsimas [6] consider the case in which each requirement $r(v,w)$ is given indirectly by node requirements $r(v)$ for each $v \in V$, where $r(v,w) = \min\{r(v), r(w)\}$. Based on a new analysis of a well-known algorithm for the minimum Steiner tree problem, they give a $\min\{2\mathcal{H}(f_{\max}), 2q\}$-approximation algorithm (for the problem in which multiple copies of each edge are allowed), where q denotes the number of distinct connectivity requirement values.

One of the main themes of the recent research in this area is the role of the primal-dual method. Agrawal, Klein, & Ravi [2] use this approach to devise a 2-approximation algorithm for the survivable network design problem in which each $r(u,v) \in \{0,1\}$. Based on this result, they obtain a $2\log_2 f_{\max}$-approximation algorithm for the survivable network design problem (where multiple copies are allowed). Goemans & Williamson [7] apply the primal-dual method to the network design problem specified by a proper $\{0,1\}$-function f to obtain a 2-approximation algorithm. The intuition for their algorithm relies heavily on the linear programming formulation of this problem, and the way in which primal-dual algorithms solve combinatorial linear programs that have integer optimal solutions. Based on this result, they derive a $2\mathcal{H}(f_{\max})$-approximation algorithm for the multiple-copy variant with arbitrary proper functions. Recently, Aggarwal & Garg [1] have strengthened this result by giving a $2\log_2 d$-approximation algorithm for the survivable network design problem, where d is the number of nodes v with $f(v) > 0$. The reader is referred to Figure 1 for a summary of the results mentioned above.

We also consider capacitated versions of the network design problem: for each edge e, at most $u(e)$ edges may be included in the solution. Note that by setting $u(e)$ to 1 for all edges, or to ∞ for all edges, we see that this definition captures all of the problems mentioned above. The capacitated problem can be reduced to the problem in which multiple copies are not allowed: replace each edge e by $\min\{f_{\max}, u(e)\}$ parallel edges. However, the algorithm no longer runs in polynomial time, and the resulting performance guarantee is rather weak. We show that a simple rounding approach yields a polynomial-time algorithm that always delivers a solution of cost within a factor of $2\mathcal{H}(\min\{f_{\max}, m\})$ of optimum, where m denotes the number of edges in G. This formulation makes the implicit assumption that the increment by which one can add capacity to an edge of the network is the same for each edge. This is rather unrealistic for some applications: when one designs a telephone network, capacity can be added to short links in small increments, whereas it is practical to add capacity to inter-continental links only in much larger increments. This motivates the following fixed-charge network design problem: the number of copies of each edge e included in the solution may be either $u(e)$ or 0. For this version, we give a polynomial-time $\min\{2f_{\max}, 2m\}$-approximation algorithm. We further consider the variant in which edges of capacity $u(e)$ may be included up to $m(e)$ times.

Our algorithm for the uncapacitated version is quite similar to the algorithm of Williamson, Goemans, Mihail, & Vazirani [14]. Both algorithms proceed in phases, where in each phase we specify cuts for which the connectivity must be augmented. The main difference between the two algorithms is the way in which these cuts are specified. In each phase of the earlier algorithm, the connectivity is augmented for *each* deficient cut (i.e., one where the requirement of the cut is not satisfied by the current partial solution), thereby reducing the maximum deficiency. In our algorithm, we augment the connectivity of only those cuts that have maximum deficiency. The analysis of each algorithm is based on bounding the additional cost of the edges selected in each phase by twice the value of a certain linear programming relaxation. The essence of the improvement in the performance guarantee is that we can prove stronger upper bounds on the optimal value of the linear programming relaxation. The earlier result bounds this value by the cost of the overall optimum solution. Consider the first phase of our algorithm; we are requiring that the connectivity be augmented by 1 only where connectivity $f_{\max}$ is required. Consequently, we can bound the linear relaxation by the overall optimum divided by $f_{\max}$. This factor of improvement is always the maximum deficiency, and hence decreases by 1 in each phase; this yields a performance guarantee of $2(1/f_{\max} + 1/(f_{\max}-1) + \cdots + 1/2 + 1) = 2\mathcal{H}(f_{\max})$.

Finally, we note that the performance guarantee of the new approach holds when our algorithm is applied to a more general class of functions f, which we call *weakly supermodular*. However, the algorithm uses a subroutine to find minimally violated subsets, and to do this in polynomial

time, one still needs the stronger assumption that f is proper.

This paper is organized as follows. In Section 2, we present some definitions and preliminary lemmas. Section 3 reviews the algorithm of Williamson et al. and shows how we can obtain the performance guarantee of $2\mathcal{H}(f_{\max})$ in the same running time. We also show that this performance guarantee is tight. Finally, Section 4 presents algorithms for capacitated versions of network design problems.

2 Preliminaries and Definitions

Formally, a *network design problem* is defined by a graph $G = (V, E)$ with n nodes and m edges, with a non-negative cost function $c : E \to \mathbf{Q}_+$ on the edges. We are also given a function $f : 2^V \to \mathbf{N}$, where $f(S)$ specifies the number of chosen edges that have to be in the cut $\delta(S)$, where $\delta(S)$ denotes the set of edges having exactly one endpoint in the set S. For any $x \in \mathbf{Q}_+^{|E|}$, define $x(\delta(S)) = \sum_{e \in \delta(S)} x(e)$. A network design problem is given by the following integer program:

$$
\begin{array}{lll}
\text{minimize} & \displaystyle\sum_{e \in E} c(e)x(e) & \\
\text{subject to:} & & \\
(IP) \qquad & x(\delta(S)) \geq f(S), & \text{for each } S \subseteq V, \\
& x(e) \in \{0, 1\}, & \text{for each } e \in E.
\end{array}
$$

In this paper, we shall impose the condition that the function f be weakly supermodular. A function $f : 2^V \to \mathbf{Z}$ is *weakly supermodular* if $f(V) = 0$, $f(S) = f(V - S)$, for each $S \subseteq V$ (symmetry), and for any two sets A and B, we have

$$f(A) + f(B) \leq \max\{f(A - B) + f(B - A), f(A \cap B) + f(A \cup B)\}.$$

We shall motivate this definition later on, by showing that it implies certain uncrossing properties of cuts. Bang-Jensen, Frank, and Jackson [3] have also introduced this class of functions, and have called them skew supermodular. As we will show below, an important special case of weakly supermodular functions are *proper* functions, as defined in Williamson et al. [14]. The function f is *proper* if $f(V) = 0$, $f(S) = f(V - S)$ for each $S \subseteq V$ (symmetry), and $f(A \cup B) \leq \max\{f(A), f(B)\}$ whenever A and B are disjoint (maximality). It is easy to show that proper implies weakly supermodular:

Lemma 2.1 Any proper function f is weakly supermodular.

Proof: By the properties of proper functions, we have the following four inequalities:

- $\max\{f(A - B), f(A \cap B)\} \geq f(A)$,
- $\max\{f(B - A), f(A \cup B)\} \geq f(A)$,
- $\max\{f(B - A), f(A \cap B)\} \geq f(B)$,
- $\max\{f(A - B), f(A \cup B)\} \geq f(B)$.

Summing the inequalities involving the minimum of $f(A-B)$, $f(B-A)$, $f(A\cap B)$ and $f(A\cup B)$, we obtain the desired result. ∎

Lemma 2.2 For any weakly supermodular function f and any $x \in \mathbf{N}^{|E|}$, the function $f(S)-x(\delta(S))$ is also weakly supermodular.

Proof: It is well known that given any $x \in \mathbf{N}^{|E|}$ and any two sets of vertices A and B,

$$x(\delta(A)) + x(\delta(B)) \geq x(\delta(A \cup B)) + x(\delta(A \cap B)),$$

and

$$x(\delta(A)) + x(\delta(B)) \geq x(\delta(A - B)) + x(\delta(B - A)).$$

The lemma statement follows straightforwardly from these inequalities. ∎

3 The Algorithm

3.1 The previous algorithm

Before discussing our algorithm for network design problems, it will first be useful to review the algorithm of Williamson et al. [14]. Their algorithm successively augments a set of edges in *phases*, starting with the empty set. At the end of phase p, the algorithm has selected a set of edges F_p that satisfies $f_p(S) = \min\{f(S), p\}$; in other words, F_p is a feasible solution to the integer program (IP) with the function $f(S)$ replaced by $f_p(S)$. Let $f_{\max} = \max_{S\subset V} f(S)$. At the end of phase $f_{\max}$, the set of edges $F_{f_{\max}}$ is a feasible solution to the original problem.

In order to find a set of edges to augment F_{p-1} to F_p, the algorithm finds a feasible solution to the following integer program:

$$\begin{array}{lll} \text{Minimize} & \sum_{e\in E} c(e)x(e) & \\ \text{subject to:} & & \\ (AUG) \qquad & x(\delta(S)) \geq h(S) & \emptyset \neq S \subset V \\ & x(e) \in \{0,1\} & e \in E_p, \end{array}$$

where

$$h(S) = \begin{cases} 1 & \text{if } f_p(S) = p \text{ and } |\delta_{F_{p-1}}(S)| < p \\ 0 & \text{otherwise,} \end{cases}$$

$E_p = E - F_{p-1}$, and $\delta_A(S)$ is defined to be $\delta(S)\cap A$ for any edge set A. If f is a proper function, then the function h is an *uncrossable* function.

Definition 3.1 A function $h : 2^V \to \{0,1\}$ is *uncrossable* if $h(V) = 0$, $h(S) = h(V-S)$ for each $V \subseteq S$, and $h(A) = h(B) = 1$ implies either $h(A\cup B) = h(A\cap B) = 1$ or $h(A-B) = h(B-A) = 1$.

Williamson et al. give an algorithm to find a feasible solution to the integer program (AUG) of cost no more than twice optimal for any uncrossable function h. The algorithm, UNCROSSABLE, takes as input a network design problem with an uncrossable function h, and returns a feasible set of edges F' satisfying the requirements defined by h. The algorithm uses the primal-dual

method for approximation algorithms, and produces a feasible solution to the dual of the linear programming relaxation of (AUG). The dual is as follows:

$$\begin{aligned} \text{Maximize} \quad & \sum_{S \in V} h(S)y(S) \\ \text{subject to:} \quad & \\ & \sum_{S: e \in \delta(S)} y(S) \leq c(e) && e \in E_p \\ & y(S) \geq 0 && S \subset V. \end{aligned}$$

UNCROSSABLE begins with the infeasible primal solution $F = \emptyset$ and the feasible dual solution $y(S) = 0$ for all S. As long as there exists a violated set with respect to h (that is, a set S such that $h(S) = 1$ and $\delta_F(S) = \emptyset$), the algorithm iteratively performs a primal-dual improvement step. In each such iteration, the algorithm first identifies all minimal violated sets for F (i.e., minimal with respect to inclusion); we will refer to these sets as *active*. Let $\mathcal{C}$ denote the collection of active sets in this iteration. The algorithm increases the value of the dual solution by uniformly raising the variables $y(C)$ corresponding to the active sets $C \in \mathcal{C}$ until the dual constraint for some edge $e \in E$ becomes tight, *i.e.*, $c(e) = \sum_{S: e \in \delta(S)} y(S)$. Edge e is then added to F, improving primal feasibility. When F becomes feasible, the algorithm executes an edge-deletion stage, eliminating unnecessary edges from F. We denote the resulting set of edges by F'. Gabow, Goemans, and Williamson [5] have shown that UNCROSSABLE can be implemented in $O(n^2 + n\sqrt{m \log\log n} + n\rho + T)$ time, where ρ is the time to compute h, and T is the total time taken to compute active sets.

The following lemma implies that UNCROSSABLE is a 2-approximation algorithm for the integer program (AUG).

Lemma 3.2 [[14]] If F' and y are the set of edges and the dual variables, respectively, returned by the algorithm UNCROSSABLE, then

$$\sum_{e \in F'} c(e) \leq 2 \sum_{S} y(S).$$

Furthermore, if ℓ_h is initial number of active sets, then

$$\sum_{e \in F'} c(e) \leq (2 - \frac{2}{\ell_h}) \sum_{S} y(S).$$

Williamson et al. show that $\sum_S y(S)$ also is a lower bound on the cost of the original network design problem. Since the final set of edges $F_{f_{\max}}$ is constructed by invoking the approximation algorithm for uncrossable functions $f_{\max}$ times, their algorithm yields a $2f_{\max}$-approximation algorithm for network design problems.

3.2 The improved algorithm

By changing the structure of the phases, we can both provide an approximation algorithm for weakly supermodular functions and dramatically improve the performance guarantee of Williamson et al. The key to our results is ensuring that in phase p, we have a set of edges F_p that satisfies $f_p(S) = f(S) - f_{\max} + p$. Notice that if f is weakly supermodular, then f_p is also weakly supermodular. Also observe that after $f_{\max}$ phases, the set of edges $F_{f_{\max}}$ is a feasible solution to the original network design problem.

In order to prove our results, we will show three things. We will first show that augmenting the set of edges F_{p-1} to the set F_p involves an uncrossable function $h_p(S) = \max\{f_p(S) -$

$|\delta_{F_{p-1}}(S)|, 0\}$, and hence we can invoke the algorithm UNCROSSABLE. Thus our algorithm for weakly supermodular functions, WEAKLY-SUPERMODULAR, consists of $f_{\max}$ invocations of UNCROSSABLE, for $p = 1, \ldots, f_{\max}$, augmenting the empty set of edges to an edge set $F_{f_{\max}}$. Second, we show that augmenting the set of edges in this fashion leads to a performance guarantee of $2\mathcal{H}(f_{\max})$. Last, we show that we can find the active sets in polynomial time given that f is a proper function. Let $m' = \min(nf_{\max}, m)$ and ρ be the time to compute f. As in [5], we find active sets in $O(n^2m' + nm'f_{\max} + n^2\rho)$ time. We prove each of these things in turn, yielding the following theorem.

Theorem 3.3 The algorithm WEAKLY-SUPERMODULAR is a $2\mathcal{H}(f_{\max})$-approximation algorithm for any weakly supermodular function f; it has $f_{\max}$ phases, and if f is proper then each phase can be implemented to run in $O(nm'f_{\max} + n^2m' + n^2\rho)$ time.

To augment a set of edges F_{p-1} to satisfy $f_p(S) = f(S) - f_{\max} + p$ given that it satisfies f_{p-1}, we must add at least one edge to each cut $\delta(S)$ for which $|\delta_{F_{p-1}}(S)| < f_p(S)$. In other words, we must find a solution to the integer program (AUG) for the function $h_p(S) = \max\{f_p(S) - |\delta_{F_{p-1}}(S)|, 0\}$. By the definition of f_p, $h_p(S) \leq 1$ for all $S \subset V$, since $|\delta_{F_{p-1}}(S)| \geq f_{p-1}(S) = f_p(S) - 1$.

Lemma 3.4 If F_{p-1} is a set of edges that satisfies the requirements defined by f_{p-1}, then $h_p(S) = \max\{f_p(S) - |\delta_{F_{p-1}}(S)|, 0\}$ is uncrossable.

Proof: Recall that the function $f_p(S) - |\delta_{F_{p-1}}(S)|$ is at most 1, and, by Lemma 2.2, it is weakly supermodular, given that f is weakly supermodular. This implies that h is uncrossable. ∎

Thus we can apply the algorithm UNCROSSABLE in $f_{\max}$ phases to construct a feasible solution for the integer program (IP). Now we show that this solution is within a factor of $2\mathcal{H}(f_{\max})$ of optimal by using the dual solutions produced by UNCROSSABLE. Consider the dual of the linear programming relaxation of the integer program (IP):

$$
\begin{array}{lll}
\text{Maximize} & \displaystyle\sum_{S \in V} f(S)y(S) - \sum_{e \in E} z(e) & \\
\text{subject to:} & & \\
 & \displaystyle\sum_{S : e \in \delta(S)} y(S) \leq c(e) + z(e) & e \in E \qquad (1) \\
(D) & y(S) \geq 0 & S \subset V \\
 & z(e) \geq 0 & e \in E
\end{array}
$$

Given the dual variables y constructed by UNCROSSABLE in phase p, define:

$$
z(e) = \begin{cases} \sum_{S : e \in \delta(S)} y(S) & \text{for all } e \in F_{p-1} \\ 0 & \text{otherwise.} \end{cases}
$$

Lemma 3.5 The vector (y, z) is a feasible solution for (D) such that $\sum_{e \in E} z(e) = \sum_S (f_p(S) - 1)y(S)$.

Proof: By the construction of y by UNCROSSABLE, for $e \in E_p = E - F_{p-1}$, we know that $\sum_{S : e \in \delta(S)} y(S) \leq c(e)$. Thus the constraints (1) hold for $e \in E_p$. For $e \in F_{p-1}$, the definition of $z(e)$ ensures that these constraints hold. This proves that (y, z) is feasible.

By the definition of $z(e)$,

$$\begin{aligned}\sum_{e\in E} z(e) &= \sum_{e\in F_{p-1}} \sum_{S:e\in\delta(S)} y(S) \\ &= \sum_{S} |\delta_{F_{p-1}}(S)| y(S) \\ &= \sum_{S} (f_p(S)-1) y(S).\end{aligned}$$

∎

Theorem 3.6 Let Z^*_{IP} be the value of an optimal solution to a network design problem given by a weakly supermodular function f, and Z^*_D be the value of an optimal solution to the dual of its linear programming relaxation. Then WEAKLY-SUPERMODULAR produces a set of edges $F_{f_{\max}}$ such that

$$\sum_{e\in F_{f_{\max}}} c(e) \le 2\mathcal{H}(f_{\max}) Z^*_D \le 2\mathcal{H}(f_{\max}) Z^*_{IP}.$$

Proof: From Lemma 3.5,

$$\begin{aligned}Z^*_{IP} &\ge \sum_S f(S)y(S) - \sum_e z(e) \\ &= \sum_S \left(f(S) - (f_p(S)-1)\right) y(S) \\ &= \sum_S \left(f(S) - (f(S) - f_{\max} + p) + 1\right) y(S) \\ &= (f_{\max} - p + 1) \textstyle\sum_S y(S).\end{aligned}$$

Using Lemma 3.2 and summing over all phases p, we obtain that

$$\sum_{e\in F_{f_{\max}}} c(e) \le \sum_{p=1}^{f_{\max}} \frac{2}{f_{\max}-(p-1)} Z^*_{IP} = 2\mathcal{H}(f_{\max})\, Z^*_{IP}.$$

∎

Finally, we must show that we can find the active sets for the uncrossable h_p in polynomial time, whenever f is proper. Recall that given a set of already selected edges F, the violated sets are the sets S such that $h_p(S) = 1$ and $\delta_F(S) = \emptyset$; given the definition of h_p, a set is violated if $|\delta_{F_{p-1}}(S)| = f_p(S) - 1$. The active sets are the minimally violated sets S. Although this cannot be done in general for weakly supermodular functions, we show how the active sets can be found when f is a proper function and thus we assume for the remainder of the section that f is proper. Our algorithm will closely follow the algorithm for finding active sets for proper f given in Gabow, Goemans, and Williamson [5]. We will need the following lemma from Williamson et al. [14].

Lemma 3.7 [[14]] Given a set of edges F, the active sets with respect to F are disjoint. Furthermore, no violated set crosses any active set.

Gabow et al. show that, given the edge set F_p, a Gomory-Hu style cut-tree H can be constructed in $O(nm'f_{\max})$ time, where $m' = \min(nf_{\max}, m)$. The edges e of the cut-tree are labeled with values $w_e \in \{0, \ldots, f_{\max}\}$. If the edge e of minimum value on the path between vertices s and t is less than $f_{\max}$, then the value of the minimum s-t cut is w_e; otherwise, it is at least $f_{\max}$. Similarly, the partition $(S_e, V - S_e)$ produced by removing e from the cut-tree has cut-value $|\delta_{F_p}(S_e)| = w_e$, unless $w_e = f_{\max}$, in which case $|\delta_{F_p}(S_e)| \geq f_{\max}$. With each edge $e = (s,t) \in H$, an associated s-t flow is constructed, which has value w_e; thus it is a max flow if $w_e < f_{\max}$. We use $\delta_H(S)$ to denote the edges of the tree H with one endpoint in the set S. Gabow et al. prove the following lemmas about the modified cut-tree.

Lemma 3.8 [[5]] For each $S \subseteq V$, $|\delta_{F_p}(S)| \geq \max_{e \in \delta_H(S)} w_e$.

Lemma 3.9 [[5]] If f is a proper function, then, for each $S \subseteq V$, $f(S) \leq \max_{e \in \delta_H(S)} f(S_e)$.

We will use the modified cut-tree to find the sets that are active at the beginning of UNCROSSABLE, and then show how this information can be updated each time the algorithm performs a primal-dual update. The following lemma shows that we can focus our attention on cuts given by the Gomory-Hu tree.

Lemma 3.10 Any violated set at the beginning of phase p is a minimum cut between s and t for some edge $e = (s,t) \in H$ satisfying $w_e = f_p(S_e) - 1$.

Proof: Let S be any violated set for F_{p-1}, i.e. $|\delta_{F_{p-1}}(S)| = f_p(S) - 1$. Since f is proper, f_p is proper, and we can apply Lemma 3.9. By the lemma, there must exist an edge $e = (s,t)$ in $\gamma(S)$ with $f_p(S_e) \geq f_p(S)$. Since F_{p-1} satisfies f_{p-1}, we must have that $w_e = |\delta_{F_{p-1}}(S_e)| \geq f_{p-1}(S_e) \geq f_p(S_e) - 1$. Lemma 3.8 implies that $w_e \leq |\delta_{F_{p-1}}(S)| = f_p(S) - 1$. Given the previous inequalities, we see that $w_e = f_p(S_e) - 1$. Since S_e is a minimum cut between s and t with value $w_e \geq f_p(S) - 1 = |\delta_{F_{p-1}}(S)|$, it follows that S is also a minimum cut between s and t. ∎

As in Gabow et al., for each edge $e = (s,t)$ of the cut-tree, we will keep track of all minimum s-t cuts by using a compact representation due to Picard and Queyranne [11]. Given a maximum flow from s to t for an edge $e = (s,t)$ in the Gomory-Hu tree, the Picard-Queyranne representation is a directed acyclic graph G_e formed from the residual graph of the flow by contracting each strongly connected component, as well as the set of all vertices reachable from s, and the set of all vertices that can reach t. For notational simplicity, let S and T denote the supervertices of G_e containing s and t respectively. Picard and Queyranne observe that there is a 1-1 correspondence between the (s,t) mincuts and the (T,S) dicuts of G_e, where a (T,S) dicut is a cut with all arcs directed from the side of the cut containing T to the side containing S. Given the s-t flow constructed for the edge (s,t), the digraph G_e can be computed in $O(m')$ time since the residual graph contains $O(m')$ edges.

Consider a topological ordering of G_e (i.e., a numbering of the supervertices so that any edge of G_e goes from a lower-numbered supervertex to a higher-numbered supervertex). Because we contracted into the supervertex T all vertices in the residual graph that can reach t, the first supervertex in the ordering can be assumed to be T. Similarly, since all vertices reachable from s have been contracted, we can assume that S is the last supervertex in the ordering. By definition, all the supervertices that are predecessors of some supervertex A in the ordering must induce a (T,S) dicut and hence an (s,t) mincut but, clearly, not all (s,t) mincuts arise in this fashion. Nevertheless, we will show that we can limit our attention to particular (s,t) mincuts arising in this way.

Lemma 3.11 Let $e = (s,t)$ be an edge of the modified cut-tree such that $w_e = f_p(S_e) - 1$. Let A be the first supervertex in a topological ordering of G_e such that $f_p(A) \geq f_p(S_e)$. Let C be the vertices of A together with those of its predecessors. If there exists an active set containing t but not s, it must be C.

Proof: Suppose there is an active set C' containing t but not s. By Lemma 3.7, an active set cannot cross a violated set and, hence, $C' \subseteq C$. Since C' is a violated set, it must be the union of supervertices of G_e. Moreover, it must contain A since $f_p(C') \geq f_p(S_e)$ and A is the only supervertex within C which has a value at least $f_p(S_e)$. Furthermore, since C' corresponds to an (S,T)-dicut of G_e, it contains all predecessors of A in G_e. Thus $C' = C$. ■

Similarly, one can find active sets containing s but not t. By doing this for all edges $e \in H$ with $w_e = f_p(S_e) - 1$, one can find in $O(nm' + n^2\rho)$ time a family of $O(n)$ violated sets guaranteed to contain all active sets. This follows from Lemma 3.10. The active sets can be extracted from the family in $O(n^2)$ time by finding the minimal sets in the family. This can be done by keeping track, for each vertex, of the set (if unique) of smallest cardinality containing it.

As in Gabow et al., each time UNCROSSABLE adds an edge e to its primal solution, the Picard-Queyranne structures can all be updated and the new potentially active sets can be found in $O(nm' + n\rho)$ time. To do this, for each residual graph we add the (bidirected) edge e, and spend $O(m')$ time recomputing its strongly connected components. At most one new component can be formed, and we evaluate f_p on this component. This takes $O(m' + \rho)$ time, and so $O(nm' + n\rho)$ time for all the n Picard-Queyranne structures.

Thus the overall time taken to find active sets in one call to UNCROSSABLE is the time taken to construct the modified cut-tree ($O(nm'f_{\max})$) plus the time taken to find the initial active sets ($O(nm' + n^2\rho)$) plus the time taken to update the active sets over the n iterations of the algorithm ($O(n^2m' + n^2\rho)$), leading to a time bound of $O(nm'f_{\max} + n^2m' + n^2\rho)$. By the previous discussion, this yields a time bound of $O(nm'f_{\max} + n^2m' + n^2\rho)$ for each call to UNCROSSABLE, and an overall running time of $O(nm'f_{\max}^2 + n^2m'f_{\max} + n^2\rho f_{\max})$ for our algorithm.

Gabow et al. show that active sets can be found in $O(nm')$ time for the survivable network design problem. By making modifications to their argument similar to those above, we can also obtain the same time bound, leading to a bound of $O(f_{\max}(nm' + n\sqrt{m\log\log n} + n\rho))$ time for the survivable network design problem.

Should this be expanded?

The performance guarantee of $O(\log f_{\max})$ is essentially tight. To show this, we construct a family of instances for which the cost of the solution returned by WEAKLY-SUPERMODULAR differs from the optimal cost by a factor exceeding $\frac{1}{2}\log_2(f_{\max})$. The network design problem in the example will be a minimum-cost flow problem, i.e., the problem of finding k edge-disjoint paths between two nodes s and t of minimum cost. The problem can be modelled by the integer program (IP) with the proper function

$$f(S) = \begin{cases} k & \text{if } |S \cap \{s,t\}| = 1 \\ 0 & \text{otherwise.} \end{cases}$$

To see that the function is proper, first note that it is obviously symmetric. Also, for any disjoint A and B, if $f(A \cup B) = k$, then $|(A \cup B) \cap \{s,t\}| = 1$ so that either $|A \cap \{s,t\}| = 1$ or $|B \cap \{s,t\}| = 1$. Thus maximality holds.

We claim that on this function f, the algorithm WEAKLY-SUPERMODULAR can be simulated by k shortest path computations. First notice that $|\delta_{F_{p-1}}(S)| \geq f_{p-1}(S)$ for all sets S with $s \in S, t \notin S$ implies that $|\delta_{F_{p-1}}(S)| \geq p - 1$. Therefore there are at least $p-1$ edge-disjoint paths between s and t. In phase p, we consider the function $h_p(S) = 1$ iff $|S \cap \{s,t\}| = 1$, and $|\delta_{F_{p-1}}(S)| = p - 1$. Choose any $p-1$ edge-disjoint paths from s to t in F_{p-1}, and direct all the

edges in the paths from t to s. Bidirect all other edges of E. Assign all directed edges in F_{p-1} a cost of 0, and all other edges their regular cost. A path directed from s to t corresponds to a solution of (AUG) of the same cost: notice that for any set S such that $s \in S$ and $h_p(S) = 1$, all zero-cost directed edges from F_{p-1} must be directed from vertices outside S to vertices in S. Similarly for S such that $t \in S$ and $h_p(S) = 1$, all zero-cost directed edges from F_{p-1} must be directed from vertices inside S to vertices outside S. Thus each set S such that $h_p(S) = 1$ is covered by an edge from $E - F_{p-1}$. Likewise, any solution to (AUG) can be made to correspond to an s-t path of no greater cost: any set S, $s \in S$, $t \notin S$, for which $h_p(S) = 0$ must have some zero-cost edge of F_{p-1} directed out of S (while any set S for which $t \in S$, $s \notin S$, and $h_p(S) = 0$ must have a zero-cost edge directed into S). This fact implies that t is reachable from s by directed edges corresponding to the solution of (AUG) plus zero-cost edges. Hence the shortest s-t path corresponds to the minimum-cost solution to (AUG). Now, since every violated set either contains s but not t or t but not s, there are two active sets initially, and $\ell_h = 2$. Applying Lemma 3.2, we obtain that UNCROSSABLE will find the minimum-cost solution to (AUG), and hence the shortest s-t path.

We can now give our tight example for WEAKLY-SUPERMODULAR on this function f. The tight example is built recursively (see Figure 2). G_0 is a graph on two nodes s and t with a single edge of cost c between them. To construct graph G_i, take two copies of G_{i-1}. Identify the nodes s and t in the two graphs. In the first copy, subdivide the unique edge of cost c entering t, and assign cost 0 to the part entering t, and c to the other part. In the second copy, subdivide the unique edge of cost c leaving s, and assign cost 0 to the part leaving s and cost c to the remaining part. Then connect the two new nodes by an edge of cost $2^{i-1}(c-\epsilon)$. The problem on graph G_i is to find $2^i - 1$ disjoint paths between s and t; we set $k = 2^i - 1$ and use the function f as above.

Given the discussion above, the algorithm WEAKLY-SUPERMODULAR will select 2^{i-1} node-disjoint paths of cost $(c-\epsilon)$, 2^{i-2} node-disjoint paths of cost $2(c-\epsilon)$, etc., for a total cost of $i2^{i-1}(c-\epsilon)$. Notice that there exists a solution of cost $(2^i-1)c$: take 2^i-1 copies of the graph G_0, where each has been subdivided into a path of length c. Hence, the cost of the solution returned by WEAKLY-SUPERMODULAR differs from the optimal cost by a factor of $\frac{i2^{i-1}}{2^i-1}\frac{c-\epsilon}{c} \geq \frac{1}{2}\log_2(f_{\max})$. This example seems to indicate that a multi-phase algorithm with a better performance guarantee needs to remove edges from previous phases.

4 Fixed-Charge Network Design

The network design problem considered in the previous section assumed that each edge can be chosen at most once; in other words, $x(e) \leq 1$. A natural generalization is to consider the case where each edge has a *capacity* $u(e)$ that bounds the number of times we can use this edge. We refer to this problem as the *capacitated network design problem.*

A natural extension of the network design problem is when the units of capacity differ from edge to edge. As an example, consider the problem of designing a telephone network. Although it is feasible to add capacity to "short" links in very small increments (say 12 telephone lines), it is infeasible to assume that the capacity of cross-continent links can be increased in the same increments (the increment there is at least several hundred lines). Roughly speaking, adding a new fiberoptic cable costs the same whether it will carry one telephone call or 1000 calls. We refer to this problem as the *fixed-charge network design problem.*

We start the discussion with the simpler capacitated network design problem. Replacing an edge with capacity $u(e)$ by $\min\{f_{\max}, u(e)\}$ unit-capacity edges and applying the algorithm WEAKLY-SUPERMODULAR we find a solution of cost within a factor of $2\mathcal{H}(f_{\max})$ of optimal. Notice,

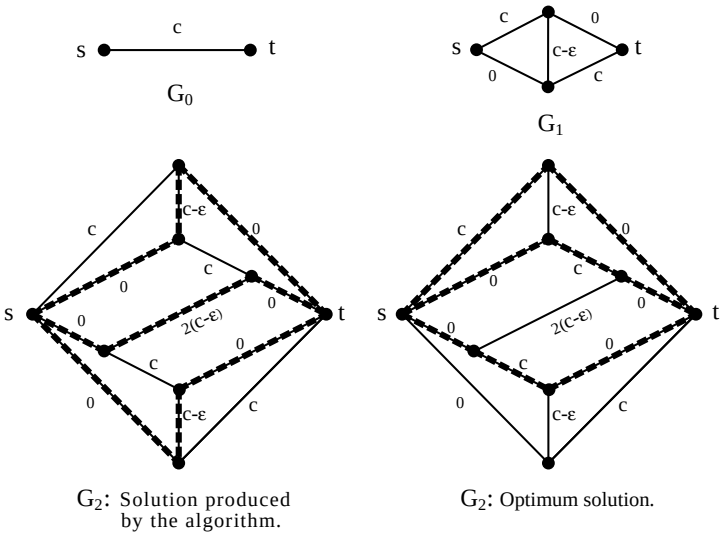

Figure 2: The tight example for WEAKLY-SUPERMODULAR. Bold edges in the first instance of G_2 are the edges selected by the algorithm. Bold edges in the second instance denote the optimal solution.

however, that when $f_{\max}$ is much larger than n, this approach has two significant problems. First, the number of phases of the algorithm is proportional to $f_{\max}$ which is not polynomial. Moreover, the performance guarantee achieved by this approach is $2\mathcal{H}(f_{\max})$, which can be rather weak.

Theorem 4.1 For any weakly-supermodular function f, there is a $2\mathcal{H}(m)$-approximation algorithm for the capacitated network design problem. If f is proper, then the algorithm can be implemented to run in polynomial time.

Proof: Consider the linear programming relaxation in which the integrality constraints on $x(e)$ are replaced by constraints $0 \leq x(e) \leq u(e)$. We use the ellipsoid method [8] or the convex programming algorithm of Vaidya [12] to solve this linear program. Note that this linear program has an exponential number of inequalities. The separation subroutine required by the linear programming algorithm is as follows: given a vector $x \in \mathbf{Q}_+^{|E|}$, find a set S such that $x(\delta(S)) < f(S)$, if one exists. Gabow et al. [5] give a polynomial time algorithm for this problem.

Let x^* denote the optimal fractional solution obtained. We include $\bar{x}(e) = \lfloor x^*(e) \rfloor$ copies of each edge e in the solution, and then apply WEAKLY-SUPERMODULAR to a reduced problem with capacities $u'(e) = u(e) - \bar{x}(e)$ and requirements $f'(S) = f(S) - \bar{x}(\delta(S))$. By Lemma 2.2, the function f' is weakly supermodular. Since x^* is feasible, for each $S \subseteq V$, $x^*(\delta(S)) \geq f(S)$, which implies that $f'_{\max} \leq m$.

Observe that $x^* - \bar{x}$ is an optimal fractional solution to the reduced problem. By Theorem 3.3, WEAKLY-SUPERMODULAR produces a solution with cost bounded by $2\mathcal{H}(f'_{\max})$ times the optimum fractional solution. Thus, the algorithm we have just described produces a solution of cost no more than $2\mathcal{H}(m)$ times the cost of x^*, the optimal fractional solution to the original problem. ∎

The *fixed-charge network design problem* can be formalized as follows. We are given a graph $G = (V, E)$, with a nonnegative cost $c(e)$ and a positive capacity $u(e)$ for each edge $e \in E$, and a weakly supermodular function f. The goal is to find a minimum-cost subset of edges such that

the total capacity of the selected edges leaving a set S is at least $f(S)$. We will assume that the capacities u and the values of the weakly supermodular function f are integral. The problem can be stated as the following integer program:

$$
\begin{array}{lll}
\text{Minimize} & \sum_{e \in E} c(e)x(e) & \\
\text{subject to:} & & \\
(CAP) & \sum_{e \in \delta(S)} u(e)x(e) \geq f(S) & \emptyset \neq S \subset V \\
& x(e) \in \{0,1\} & e \in E.
\end{array}
$$

In the case that all capacities $u(e) = u$ for some fixed positive integer u and the function f is proper, the problem reduces to the network design problem considered in Section 3, with the proper function $f'(S) = \lceil f(S)/u \rceil$. Applying WEAKLY-SUPERMODULAR, we get a $2\mathcal{H}(\lceil f_{\max}/u \rceil)$-approximation algorithm. Notice that this observation doesn't apply to weakly supermodular functions f since the corresponding function f' might not be weakly supermodular.

If the capacities are not all the same, then we apply APPROXIMATE-UNCROSSABLE in a sequence of at most $f_{\max}$ phases. Let F_{i-1} denote the set of edges at the end of phase $i-1$. Initially, $F_0 = \emptyset$. If F_{i-1} is feasible then we stop. Otherwise, consider the remaining deficiency function $f_i(S) = f(S) - u(\delta_{F_{i-1}}(S))$, where we use $u(A)$ to denote $\sum_{e \in A} u(e)$. Let $k_i = \max_S f_i(S)$. As a step towards feasibility, phase i finds a set of edges F_i' which cover all the cuts of maximum deficiency. More precisely, in phase i, we apply UNCROSSABLE to the instance with costs $c(e)$, edge set $E_i = E - F_i$, and uncrossable function:

$$
h_i(S) = \begin{cases} 1 & \text{if } f_i(S) = k_i \\ 0 & \text{otherwise.} \end{cases}
$$

The resulting set of edges, F_i', is then added to F_{i-1} to give F_i. By Lemma 2.2, the function f_i is weakly supermodular, which implies that h_i is uncrossable. This justifies our application of UNCROSSABLE.

Theorem 4.2 The algorithm described above is a $2\min\{m, f_{\max}\}$-approximation algorithm for the fixed-charge network design problem. It terminates after $\min\{m, f_{\max}\}$ phases, and if the function f is proper then each phase can be implemented to run in $O(f_{\max} n^2 m' + n^2\rho)$ time, where $m' = \min\{m, nf_{\max}\}$, and ρ is the time required to compute $f(S)$ for a set S.

Proof: It is easy to see that $k_{i+1} < k_i$ and that E_{i+1} is a proper subset of E_i; this implies the bound on the number of phases.

In the previous section, we have shown how to implement this algorithm for proper functions; the same argument can also be used when the function is the difference of a proper function and a degree function, as is f_i. This leads to an implementation of a phase that runs in $O(f_{\max} n^2 m' + n^2\rho)$ time.

We now prove the performance guarantee. In phase i, we incur a cost of $\sum_{e \in F_i'} c(e)$. By Lemma 3.2, we know that $\sum_{e \in F_i'} c(e) \leq 2Z_i^*$, where Z_i^* denotes the optimum value of the integer program corresponding to the uncrossable function h_i.

Let x^* be an optimal solution to (CAP), with value Z_{IP}^*. Observe that by assigning $x^*(e)$ for each $e \in E_i$, we get a feasible solution to the integer program with requirements h_i and edge set E_i. Hence, $Z_i^* \leq Z_{IP}^*$. Combined with the previous inequality, this shows $\sum_{e \in F_i'} c(e) \leq 2Z_{IP}^*$.

In other words, the cost incurred in every phase is bounded by twice the optimal value. This, together with the fact that the number of phases is bounded by $\min\{m, f_{\max}\}$, implies the claim of the theorem. ∎

If $f_{\max}$ is large, the performance guarantee of the above algorithm is $2m$. It is not hard to see that a simple greedy algorithm has performance guarantee m. Consider the edges in order of decreasing cost, and greedily delete an edge if the remaining graph satisfies the requirements. We claim that this greedy algorithm is an m-approximation algorithm. Let e be the first edge on which the greedy solution differs from the optimal solution, let C be the cost of the edges shared by the greedy and the optimal solution. Then the cost of the optimal solution is at least $C + c(e)$, whereas the cost of the greedy solution is at most $C + mc(e)$.

Using techniques analogous to the proof of Theorem 4.1 the algorithms and the guarantees of Theorem 4.2 extend to a further generalization of the model when multiple edges are allowed up to a limited number of multiplicities. This problem is modeled by replacing the restriction expand? $x(e) \in \{0, 1\}$ in the integer program (CAP) by restrictions of the form $x(e) \in \{0, 1, \ldots, m(e)\}$.

Our algorithm for the fixed-charge network design problem can be used to solve a network reinforcement problem recently investigated by Bienstock and Diaz [4]. Given a graph $G = (V, E)$, edge costs $c(e)$, and a weakly supermodular function f, one must find a minimum-cost set of edges such that the graph induced by contracting these edges satisfies f. In other words, the set must include one edge from every set S such that $|\delta(S)| < f(S)$. By using the function $f'(S) = f(S) - |\delta(S)|$ and setting $u(e) = f'_{\max}$ for all e, we can apply the algorithm above.

We give an example that shows that the performance guarantee of $O(f_{\max})$ of our algorithm is asymptotically tight; see Figure 3. Let G be a graph on $k + 2$ nodes: s, t, and v_i for $i = 1, \ldots, k$. For each $i = 1, \ldots, k$, there is an edge (s, v_i) of cost $c(s, v_i) = 0$; for $i \neq k$, it has capacity $u(s, v_i) = 1$ whereas the capacity $u(s, v_k) = k$. For each $i = 1, \ldots, k$, there is an edge (v_i, t) of capacity $u(v_i, t) = k$; for $i \neq k$, it has capacity $c(v_i, t) = c$ whereas the capacity $c(v_k, t) = c + \epsilon$. The proper function f is k for every s-t cut and 0 otherwise. As we have mentioned in the previous section, UNCROSSABLE finds the minimum-cost augmenting path from s to t with cost 0 on backward edges, and costs c_i on forward edges. Therefore, phase i selects $F'_i = \{(s, v_i), (v_i, t)\}$. This results in a solution of value $kc + \epsilon$, whereas the optimum solution is $\{(s, v_k), (v_k, t)\}$ at a cost of $c + \epsilon$. It is interesting to note that for this fixed-charge problem, we do not know a better algorithm even for the special case considered in the example: the case when there is only one pair of nodes with a positive connectivity requirement.

We can improve this bound if there is no upper bound on the multiplicity with which an edge can be included. More precisely, the restriction $x(e) \in \{0, 1\}$ in (CAP), is replaced by $x(e) \geq 0$ and integral, for each edge e. We give an $4 \log_2 f_{\max}$-approximation algorithm for the case when f is a proper function.

The algorithm is analogous to that of Agrawal, Klein and Ravi [2] and Goemans and Williamson [7] for the analogous version of the network design problem where multiple edges are allowed. The approach can be thought of as being halfway between the approach of Williamson et al. [14] and the present paper. In the first phase we consider sets with deficiency more than $f_{\max}/2$, and take a sufficient number of copies of the edges of the resulting forest, so as to decrease the maximum deficiency to at most $f_{\max}/2$.

In phase i, we find a set of edges F'_i that augment F_{i-1} to F_i. We first construct the same capacitated network design instance as was done in phase i of the previous algorithm. However, in this case we apply the algorithm UNCROSSABLE with costs $c_i(e) = c(e)/u(e)$ and the requirement

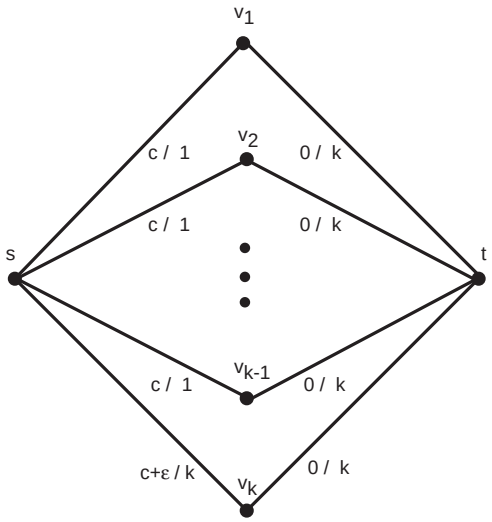

Figure 3: Tight example for the fixed-charge network design algorithm. Edges are labelled with the edge cost/edge capacity.

function

$$h_i(S) = \begin{cases} 1 & \text{if } f_i(e) \geq k_i/2 \\ 0 & \text{otherwise.} \end{cases}$$

We include each edge e in the resulting set of edges F' with multiplicity $\lceil \frac{k_i}{2u(e)} \rceil$.

To apply algorithm UNCROSSABLE we need to argue that the function h is indeed uncrossable. Williamson et al. [14] proved that if f is proper and x is a vector on the edges, then, for any constant r, the following function h is uncrossable: $h(S) = 1$ if $f(S) - x(\delta(S)) \geq r$, and $h(S) = 0$ otherwise. Notice that this statement is not true for weakly supermodular functions in general. The proof of the following theorem is completely analogous to the proof of Theorem 4.2.

Theorem 4.3 For any proper function f, the algorithm described above is a $4\lceil \log_2 k \rceil$-approximation algorithm for the fixed-charge network design problem where multiple copies of edges are allowed. It terminates after $\lceil \log_2 f_{\max} \rceil$ phases and each phase can be implemented to run in $O(kn^2m' + n^2\rho)$ time, where $m' = \min\{m, nf_{\max}\}$, and ρ is the time required to compute $f(S)$ for a set S.

Proof: We have already shown that function h is uncrossable, and hence algorithm UNCROSSABLE applies. One phase decreases the maximum deficiency k_i by a factor of 2. This implies the bound on the number of phases. The running time of one phase is the same as in the proof of Theorem 4.2.

Consider the cost incurred by one iteration of the algorithm. We proceed analogous to the proof of Theorem 4.2. The network design problem considered in the ith iteration with $u(e)$ parallel copies for every edge $e \in E_i$ with cost $c_i(e) = c(e)/u(e)$ each. Notice that the minimum cost Z_i^* of the capacitated network design problem considered in each iteration is a lower bound on the cost of original problem.

Analogous to the proof of Theorem 3.6 we can show that the cost of the edges F_i' selected in iteration i is bounded by

$$c_i(F_i') \leq \frac{4}{k_i} Z_i^*.$$

To bound the cost incurred during phase i notice that every edge $e \in F_i'$ is taken with multiplicity so that the resulting capacity is between $k_i/2$ and k_i. Therefore, the cost incurred is at most $4Z_i^*$. The number of iterations is bounded by $\lceil \log_2 k \rceil$. This proves the claimed performance guarantee. ∎

References

[1] M. Aggarwal and N. Garg. A scaling technique for better network design. In *Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 233–240, 1994.

[2] A. Agrawal, P. Klein, and R. Ravi. When trees collide: An approximation algorithm for the generalized Steiner problem on networks. *SIAM Journal on Computing*, 24:440–456, 1995.

[3] J. Bang-Jensen, A. Frank, and B. Jackson. Preserving and increasing local edge-connectivity in mixed graphs. *SIAM Journal on Discrete Mathematics*, 8, 1995.

[4] D. Bienstock and N. Diaz. Blocking small cuts in a network, and related problems. *SIAM Journal on Computing*, 22:482–499, 1993.

[5] H. N. Gabow, M. X. Goemans, and D. P. Williamson. An efficient approximation algorithm for the survivable network design problem. In *Proceedings of the Third MPS Conference on Integer Programming and Combinatorial Optimization*, pages 57–74, 1993. To appear in *Math. Programming*.

[6] M. X. Goemans and D. J. Bertsimas. Survivable networks, linear programming relaxations and the parsimonious property. *Mathematical Programming*, 60, 1993. Also appears as MIT Tech Report OR 216-90.

[7] M. X. Goemans and D. P. Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24:296–317, 1995.

[8] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithm and Combinatorial Optimization*. Springer-Verlag, Berlin, 1988.

[9] P. Klein and R. Ravi. When cycles collapse: A general approximation technique for constrained two-connectivity problems. In *Proceedings of the Third MPS Conference on Integer Programming and Combinatorial Optimization*, pages 39–55, 1993. Also appears as Brown University Technical Report CS-92-30.

[10] T. L. Magnanti and R. T. Wong. Network design and transportation planning: Models and algorithms. *Transportation Science*, 18:1–55, 1984.

[11] J.-C. Picard and M. Queyranne. On the structure of all minimum cuts in a network and applications. *Mathematical Programming Study*, 13:8–16, 1980.

[12] P. M. Vaidya. A new algorithm for minimizing convex functions over convex sets. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pages 338–343, 1989.

[13] D. P. Williamson. *On the design of approximation algorithms for a class of graphs problems*. PhD thesis, MIT, Cambridge, MA, September 1993.

[14] D. P. Williamson, M. X. Goemans, M. Mihail, and V. V. Vazirani. A primal-dual approximation algorithm for generalized Steiner network problems. *Combinatorica*, 15:435–454, 1995.