A Primal-Dual Approximation Algorithm for Generalized Steiner Network Problems

*David P. Williamson** M.I.T.
Michel X. Goemans† M.I.T.
Milena Mihail Bellcore
Vijay V. Vazirani‡ I.I.T., Delhi

Abstract

We present the first polynomial-time approximation algorithm for finding a minimum-cost subgraph having at least a specified number of edges in each cut. This class of problems includes, among others, the generalized Steiner network problem, also called the survivable network design problem. If k is the maximum cut requirement of the problem, our solution comes within a factor of $2k$ of optimal. Our algorithm is primal-dual and shows the importance of this technique in designing approximation algorithms.

1 Introduction

We consider the class of problems of finding a minimum-cost subgraph such that the number of edges crossing each cut is at least a specified requirement, which is some function of the cut. More formally, given an undirected graph $G = (V, E)$, a non-negative cost function $c : E \rightarrow \mathbb{Q}_+$, and a function $f : 2^V \rightarrow \mathbb{N}$, this class of problems can be formulated as the following integer program:

$$\text{Min} \quad \sum_{e \in E} c_e x_e$$

subject to:

$$(IP) \qquad \sum_{e \in \delta(S)} x_e \geq f(S) \qquad \emptyset \neq S \subset V$$
$$x_e \in \{0, 1\} \qquad e \in E$$

where $\delta(S)$ denotes the set of edges having exactly one endpoint in S.

In this paper, we shall impose the condition that the function f is *proper*, i.e. f satisfies:

- [Symmetry] $f(S) = f(V - S)$ for all $S \subseteq V$; and
- [Maximality] If A and B are disjoint, then $f(A \cup B) \leq \max\{f(A), f(B)\}$.

We shall motivate this definition later on, by showing that it implies certain uncrossing properties of cuts.

The formulation (IP) captures a vast class of graph connectivity problems. For example, it encompasses the problem of finding a *minimum-cost k-edge-connected subgraph* (by using $f(S) = k$, for all $\emptyset \subset S \subset V$, and consequently ensuring that there are k edge-disjoint paths between any pair of vertices i and j). It also encompasses a well-known generalization of both the k-edge-connectivity problem and the Steiner tree problem called *the generalized Steiner network problem* or the *survivable network design problem*. In this problem, given requirements r_{ij} for each pair i, j of vertices, we need to find a minimum-cost subgraph that has r_{ij} edge-disjoint paths between i and j. This problem is modelled by (IP) with the function $f(S) = \max_{i \in S, j \notin S} r_{ij}$, which can be easily seen to be proper. This problem has a number of important practical applications. It comes up, for instance, in the design of networks which can "survive" certain edge failures (e.g. see [7]).

Solving (IP) optimally is well known to be NP-hard even under many restrictions. For example, the Steiner tree problem is NP-hard even when the cost function satisfies the Euclidean metric, and the minimum 2-edge connected subgraph problem is NP-hard even if all edge weights are unity. Efficient approxi-

*Research supported by an NSF Graduate Fellowship, DARPA contracts N00014-91-J-1698 and N00014-92-J-1799, and AT&T Bell Laboratories.

†Research supported in part by Air Force contract F49620-92-J-0125 and DARPA contract N00014-92-J-1799.

‡Part of this work was done while the author was visiting AT&T Bell Laboratories and Bellcore.

25th ACM STOC '93-5/93/CA,USA
© 1993 ACM 0-89791-591-7/93/0005/0708...$1.50

mation algorithms for (IP) were known only for very special cases.

Our main result is the first polynomial time approximation algorithm for (IP). Our algorithm finds a solution within a factor of $2k$ of the optimal, where $k = \max_S f(S)$. We should stress that, in our model, each edge is used at most once. The running time is dominated by $O(k|V|^3)$ maximum flow computations, given that f is an oracle to which the algorithm has free access. This running time has since been improved by Gabow, Goemans, and Williamson [4].

If the function f takes only l distinct values $0 = \rho_0 < \rho_1 < \rho_2 < \ldots < \rho_l$, then the performance guarantee is improved to $2\sum_{i=1}^{l} \mathcal{H}(\rho_i - \rho_{i-1})$, where $\mathcal{H}$ is the harmonic function $\mathcal{H}(k) = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{k}$. In particular, for the minimum k-edge-connected subgraph problem (or its Steiner version where only a subset of the vertices need to be k-edge-connnected), the approximation factor is $2\mathcal{H}(k) \sim 2\ln k$. Furthermore, when $\rho_1 = 1$ and $l \geq 2$, the bound can be improved to $2\sum_{i=1}^{l} \mathcal{H}(\rho_i - \rho_{i-1}) - 1$. Our algorithm immediately extends to augmentation versions of these problems, where one needs to *augment* a given graph so as to satisfy a proper function f at minimum cost. The algorithm has the same performance guarantee, or sometimes even better if the given graph partially meets the connectivity requirements.

Perhaps the most significant aspect of this work lies in the methodology underlying our algorithm, which is based on a primal-dual approach. The power of this approach has been utilized extensively for solving problems in P. In fact, the most efficient known algorithms for some of the cornerstone problems in combinatorial optimization, including matching, flow and shortest paths, are based on this approach [9]. Some of these algorithms, e.g., Dijkstra's shortest path algorithm, can be described without referring to the primal-dual framework. In other cases, though, such as Edmonds' algorithm for the minimum-weight nonbipartite matching problem, the primal-dual approach appears to be crucial. The method consists of performing alternate primal and dual improvement steps until an optimal solution is found. The special combinatorial structure of the particular problem is used for designing procedures for the improvement step, thus resulting in an efficient algorithm.

The primal-dual approach can also be used to derive good approximation algorithms for NP-hard optimization problems; in this context, the primal and dual improvement steps attempt at homing in on a "good" approximate integral solution. Many known approximation algorithms can be interpreted as primal-dual. Goemans and Williamson [6], motivated by an earlier result of Agrawal, Klein, and Ravi [1], show that the primal-dual technique can be applied to a large class of graph problems. They present a 2-approximation algorithm for (IP), provided f ranges in $\{0,1\}$, by using an algorithm that successively improves primal and dual solutions to the LP relaxation of (IP). Furthermore, for several NP-hard problems the primal-dual algorithm in [6] is the only known method yielding polynomial-time approximations with good performance guarantees. On the other hand, for optimization problems in P, the primal-dual approach is simply an efficient method for solving the corresponding linear programs. For this reason, the primal-dual approach seems to hold even more promise in the context of approximation algorithms than in the design of exact algorithms. In the next section we present elements of the primal-dual approach to approximation algorithms. This will also help state the high level idea behind our algorithm for approximating (IP).

In addition to the primal-dual approach, our algorithm uses the idea of satisfying f in "phases". This idea was introduced by Ravi and Klein [10] who gave the first 3-approximation algorithm for proper functions with range $\{0,2\}$. Their approach, though, does not seem to extend to the general case since their analysis relies heavily on properties of 2-edge-connected graphs.

The only previous related approximation results that we are aware of are the following. For the Steiner tree problem (the case where $r_i \in \{0,1\}$ for all i, and $r_{ij} = \min(r_i, r_j)$), the previously best bound of 2 was first improved by Zelikovsky [12] and then refined to 16/9 by Berman and Ramaiyer [2]. Goemans and Bertsimas [5] showed an approximation algorithm for the generalized Steiner network problem where $r_{ij} = \min(r_i, r_j)$ and multiple copies of edges are allowed. Agrawal, Klein, and Ravi [1] derived a 2-approximation algorithm for the generalized Steiner tree problem ($r_{ij} \in \{0,1\}$) and used it to derive an approximation algorithm for the generalized Steiner network problem in the case where multiple copies of edges are allowed. Goemans and Williamson [6] generalized this result to apply to all proper functions with range $\{0,1\}$. Saran, Vazirani and Young [11] applied the technique of Goemans and Williamson to the 2-edge-connected problem, achieving the same guarantee as an earlier algorithm of Frederickson and Ja'Ja' [3]. Ravi and Klein [10] generalized the work of Goemans and Williamson to proper functions with range $\{0,2\}$. For k-edge-connectivity problems, Khuller and Vishkin [8] have shown a very simple 2-approximation algorithm for the k-connected subgraph problem. However, their algorithm does not seem to extend even to the case where $r_i \in \{0,k\}$ and $r_{ij} = \min(r_i, r_j)$. Notice that even for generalized Steiner network problems with $r_{ij} \in \{0,1,2\}$ no

polynomial-time approximation algorithm was known prior to our work.

The rest of the paper is structured as follows. In Sections 2 and 3, we present the primal-dual method for approximation algorithms and give a detailed presentation of our algorithm. We prove that the algorithm runs in polynomial time in Section 4. Sections 5 and 6 give a proof of its performance guarantee. We conclude with a few remarks in Section 7.

2 The Primal-Dual Method for Approximation Algorithms

A fundamental fact of linear programming duality is that a primal feasible and a dual feasible solution are both optimal if and only if they satisfy all the complementary slackness conditions. For problems in P, the primal-dual method starts with arbitrary primal infeasible and dual feasible solutions and iteratively improves the value of the dual solution and the feasibility of the primal solution. Throughout the execution, the complementary slackness conditions are imposed and guide the improvement steps. The algorithm stops as soon as the primal solution is feasible. The two solutions evolve hand-in-hand; improvements to the primal solution are based on the current dual solution, and vice versa.

When applied to approximation algorithms, the primal-dual method appears to allow for a richer range of operating mechanisms. For concreteness, consider the LP relaxation of (IP) stated in the introduction, under the assumpion that f ranges in $\{0,1\}$, i.e., the original Goemans and Williamson setting [6]. This setting is simpler than ours because in the LP relaxation the variables x_e do not need to be explicitly bounded above by 1. The LP relaxation (primal) and its dual are:

$$\begin{array}{lll} & \text{Min } \sum_{e\in E} c_e x_e & \\ s.t. & \sum_{e:e\in\delta(S)} x_e \geq f(S) & S \subset V \\ & x_e \geq 0 & e \in E \end{array}$$

$$\begin{array}{lll} & \text{Max } \sum_S f(S) y_S & \\ s.t. & \sum_{S:e\in\delta(S)} y_S \leq c_e & e \in E \\ & y_S \geq 0 & S \subset V. \end{array}$$

Notice that the dual LP is seeking an optimal packing of cuts.

Since we wish to find an integral solution to the primal, obviously we should not be trying to satisfy all the complementary slackness conditions. These conditions are of two types:

(a). Primal complementary slackness conditions:

Input: An undirected graph $G = (V, E)$, edge costs $c_e \geq 0$, a proper function f, and $k = \max_S f(S)$
Output: A set of edges F_k

1 $F_0 \leftarrow \emptyset$
2 For $p \leftarrow 1$ to k
3 *Comment: Phase p.*
4 $g_p(S) = \min\{f(S), p\}$ for all $S \subset V$.
5 $h(S) = 1$ if $g_p(S) = p$ and $|\delta_{F_{p-1}}(S)| = p - 1$; $h(S) = 0$ otherwise.
6 $E_h \leftarrow E - F_{p-1}$
7 Apply algorithm for uncrossable functions (Fig. 2) to $G = (V, E_h)$, edge costs c_e, and function h, yielding F'
8 $F_p \leftarrow F_{p-1} \cup F'$
9 Output F_k

Figure 1: The approximation algorithm for (IP).

these correspond to the primal variables:

$$\forall e : x_e > 0 \Rightarrow \sum_{S:e\in\delta(S)} y_S = c_e.$$

(b). Dual complementary slackness conditions: these correspond to the dual variables:

$$\forall S : y_S > 0 \Rightarrow \sum_{e\in\delta(S)} x_e = f(S).$$

Primal-dual approximation algorithms generally operate by ensuring the first set of conditions and relaxing the second set to

(b'). $\quad \forall S : y_S > 0 \Rightarrow f(S) \leq \sum x_e \leq \alpha f(S),$

for some constant α. Thus the primal solution found is within a factor of α of the optimal primal LP solution, and therefore also within a factor of α of the optimal solution to (IP).

The primal-dual algorithm of Goemans and Williamson is somewhat different. Initially it starts with the feasible dual solution $y = 0$, and its dual improvement step only increases the y variables. Unlike other primal-dual algorthms it ensures conditions (b') only when averaged over the sets S whose dual variables y_S are being raised during a given dual improvement step.

The situation with our algorithm is significantly more involved. We do not know how to find an integral solution to (IP) directly, as the LP relaxation is more complicated because of inequalities ensuring that each edge is selected at most once. Instead, we decompose the task into k phases, where $k = \max_S f(S)$, as follows (see also Figure 1). In phase p, define the p^{th} truncation of a proper function f to be the function $g_p(S) = \min\{f(S), p\}$. Notice that g_p is also a proper function. We shall ensure that the union of edges chosen in the first p phases, F_p, satisfies g_p in the sense

that for all $S \subset V$, $|\delta_{F_p}(S)| \geq g_p(S)$, where $\delta_A(S)$ is defined as $\delta(S) \cap A$.

In order to augment the edge set F_{p-1} to a set F_p satisfying g_p, we need to cover all cuts $\delta(S)$ for which $g_p(S) = p$ and $|\delta_{F_{p-1}}(S)| = p-1$ with edges from $E - F_{p-1}$. For convenience, we define $h : 2^V \rightarrow \{0,1\}$ to be the function that sets $h(S) = 1$ iff $g_p(S) = p$ and $|\delta_{F_{p-1}}(S)| = p-1$. This function has the following interesting uncrossing property, which we will prove in Theorem 3.5 in the next section. If $h(A) = h(B) = 1$ then either $h(A-B) = h(B-A) = 1$ or $h(A \cap B) = h(A \cup B) = 1$. Any function with these properties will be called an *uncrossable function*. Now, in the p^{th} phase, the algorithm chooses edges from $E_h = E - F_{p-1}$ so as to satisfy h.

The minimum-cost way of augmenting F_{p-1} to F_p can be formulated as an integer program, (IP_h). In the p^{th} phase, our algorithm finds a "good" approximate solution to (IP_h) and a corresponding packing of cuts for the dual of the LP relaxation:

$$\begin{array}{lll} (IP_h) & \text{Min } \sum_{e \in E_h} c_e x_e & \\ s.t. & \sum_{e \in \delta(S)} x_e \geq h(S) & S \subset V \\ & x_e \in \{0,1\} & e \in E_h \end{array}$$

$$\begin{array}{lll} & \text{Max } \sum_S h(S) y_S & \\ s.t. & \sum_{S: e \in \delta(S)} y_S \leq c_e & e \in E_h \\ & y_S \geq 0 & S \subset V. \end{array}$$

An approximation algorithm for (IP_h) for any uncrossable function h is described in the following section. In much the same way as [6] it finds a set of edges F' that satisfies h and a feasible dual solution y such that the primal complementary slackness conditions will be enforced exactly, and the dual ones will be relaxed, and enforced in an average sense (for $\alpha = 2$). We will derive the following key lemma.

Lemma 2.1 [Main Lemma] Let F' be the set of edges and y the dual feasible solution constructed by the algorithm for (IP_h). Then

$$\sum_{e \in F'} c_e \leq \left(2 - \frac{2}{\ell}\right) \sum_{S \subset V} h(S) y_S,$$

where ℓ is the maximum number of disjoint sets S of V for which $h(S) = 1$.

The proof of the lemma is more involved than the proof of the corresponding lemma in [6]. We further show that the dual solution found can be transformed into a feasible dual solution for the linear programming relaxation of (IP) of at least the same value. Therefore, the edges picked in the p^{th} phase F' have weight within $2 \sum_{S \subset V} h(S) y_S \leq 2 \cdot Z^*_{IP}$, where Z^*_{IP} is the cost of the optimal solution to (IP). Summing over all phases, we obtain a solution to (IP) which is within $2k \cdot Z^*_{IP}$.

3 Approximation Algorithm for an Uncrossable Function

In this section, we describe an approximation algorithm for (IP_h) for any uncrossable function h. As stated in the previous section, the use of phases reduces the problem of approximating (IP) to approximating (IP_h) for a particular uncrossable function h on the edge set $E_h = E - F_{p-1}$ in each phase p. We show that these functions h are uncrossable at the end of the section.

Given an uncrossable function h and edge set E_h, the algorithm consists of two stages. In the first stage, the algorithm starts with an empty forest F and iteratively adds edges until the resulting forest F is feasible. In this context, feasibility means that there are no "violated sets" with respect to h, where a *violated set* is a set S with $h(S) = 1$ and $\delta_F(S) = \emptyset$. In the second stage the algorithm deletes redundant edges.

A central fact that is used both by the algorithm and in its analysis is that the minimal violated sets with respect to h are disjoint. By a "minimal" violated set, we mean that none of its proper subsets are violated.

Theorem 3.1 Let h be an uncrossable function and let F be any subset of E_h. Then the minimal violated sets with respect to h are disjoint.

Proof: Suppose A, B are minimal violated sets and A and B are not disjoint. Since A, B are violated sets, $h(A) = h(B) = 1$. By definition of an uncrossable function, we know that either $h(A-B) = h(B-A) = 1$ or $h(A \cup B) = h(A \cap B) = 1$. Suppose that the second case holds (the proof of the other case is similar). By submodularity of $|\delta_F(S)|$, we have

$$|\delta_F(A)| + |\delta_F(B)| \geq |\delta_F(A \cap B)| + |\delta_F(A \cup B)|.$$

Our assumption that A and B are violated implies that $|\delta_F(A)| = |\delta_F(B)| = 0$. Therefore, $|\delta_F(A \cap B)| = |\delta_F(A \cup B)| = 0$ and, hence, $A \cap B$ and $A \cup B$ are violated sets. This contradicts the minimality of A and B, however, so that it must be the case that A and B are disjoint. ■

We are now ready to describe the algorithm for approximating (IP_h) in more detail (see also Figure 2). In the first stage, the algorithm begins with the primal infeasible solution $F = \emptyset$ and the dual feasible solution $y_S = 0$ for all S. As long as there exist violated sets with respect to h (that is, the primal solution is infeasible), the algorithm iteratively performs a primal-dual improvement step. In such a step, the algorithm first identifies all minimal violated sets for F. We will refer to these sets as *active*. Let $\mathcal{C}$ denote the collection of active sets in this iteration. The algorithm then

uniformly raises the variables y_C corresponding to the active sets $C \in \mathcal{C}$ until the dual constraint for some edge $e \in E_h$ becomes tight, i.e.,

$$c_e = \sum_{S:e \in \delta(S)} y_S.$$

Edge e is then added to F. If F becomes feasible for (IP_h), the algorithm goes on to the second stage, otherwise it iterates the primal-dual improvement step. At the end of the first stage, the algorithm has both a primal and a dual feasible solution for (IP_h) such that the primal complementary slackness conditions hold (that is, $c_e = \sum_{S:e \in \delta(S)} y_S$ for all $e \in F$).

The second stage of the algorithm, called *reverse delete*, deletes redundant edges from F. As will prove crucial, we consider edges for deletion in the reverse of the order in which they were added to F. When edge e is considered, it is removed from the current set F if $F - e$ still satisfies h. At the end of the reverse delete stage, the remaining set of edges, F', is still primal feasible and the primal complementary slackness conditions still hold. In addition, the dual complementary slackness conditions will now hold in the average sense described in the previous section. Proving this fact will result in the proof of the Main Lemma, as will be shown in Section 5.

The algorithm is formally described in Figure 2. As can be seen from the figure, the dual variables y need not be explicitly maintained. Instead, we only need to keep track of the variables $d(v)$, which can be shown to be equal to $\sum_{S:v \in S} y_S$ at the beginning of each iteration of the first stage. The ability to implement this algorithm in polynomial time crucially depends on whether the active sets can be found efficiently. In the next section, we show that it is possible to find these sets for uncrossable functions that arise in our algorithm for (IP). Other implementation issues (such as finding the next edge to add to F) can be handled efficiently as in Goemans and Williamson [6].

It remains to show that the function h as defined in the algorithm for (IP) is in fact uncrossable. We do this in Theorem 3.5 below.

Observation 3.2 Let f be a proper function, and let A, B, C form a partition of V. Then the maximum of $f(A)$, $f(B)$, and $f(C)$ is not uniquely attained.

Proof: Let C attain the maximum. The observation follows from the symmetry property applied to $V - C$ and the maximality property applied to A, B, and $A \cup B = V - C$. ■

Corollary 3.3 Let f be a proper function. For disjoint sets A and B, the maximum of $f(A)$, $f(B)$, and $f(A \cup B)$ is not uniquely attained.

Input: An undirected graph $G = (V, E_h)$, edge costs $c_e \geq 0$, and an uncrossable function h
Output: A set of edges F'

```
1   F ← ∅
2   Comment: Implicitly set y_S ← 0 for all S ⊂ V
3   i ← 0
4   d(v) ← 0 for all v ∈ V
5   C ← all active sets C (minimal sets not satisfying h(S)).
6   While |C| > 0
7     i ← i + 1
8     Comment: Begin iteration i.
9     For all v ∈ C ∈ C, increase d(v) uniformly by ε until
      some edge e_i = (u, v) ∈ E_h and e_i ∈ δ(C) for some
      C ∈ C satisfies d(u) + d(v) = c_uv.
10    Comment: Implicitly set y_C ← y_C + ε for all C ∈ C.
11    F ← F ∪ {e_i}
12    Update C
13    Comment: End iteration i.
14  Comment: Reverse delete stage
15  F' ← F
16  For j ← i downto 1
17    If F' − {e_j} satisfies h
18      F' ← F' − {e_j}
19  Output F'
```

Figure 2: The algorithm for uncrossable functions h.

Lemma 3.4 Let f be a proper function. Let A and B be sets of vertices such that $f(A) \geq p$ and $f(B) \geq p$. Then, either $f(A - B) \geq p$ and $f(B - A) \geq p$, or $f(A \cap B) \geq p$ and $f(A \cup B) \geq p$.

Proof: Assume for example that $f(A-B) < p$. Then, by applying the corollary above to $A - B$, $A \cap B$, and A, we see that $f(A \cap B) \geq p$. Applying the corollary to $A - B$,B, and $A \cup B$ gives $f(A \cup B) \geq p$. The other cases can be treated similarly. ■

Theorem 3.5 Let F_{p-1} be a set of edges such that $|\delta_{F_{p-1}}(S)| \geq g_{p-1}(S)$ for all $S \subseteq V$. Let $h(S) = 1$ if $g_p(S) = p$ and $|\delta_{F_{p-1}}(S)| = p - 1$, and let $h(S) = 0$ otherwise. Then h is an uncrossable function.

Proof: Let A and B be sets of vertices such that $h(A) = h(B) = 1$. Since g_p is a proper function, Lemma 3.4 implies that either $g_p(A \cup B) = g_p(A \cap B) = p$ or $g_p(A-B) = g_p(B-A) = p$. Suppose that the first case holds. Since F_{p-1} satisfies g_{p-1}, we must have that $|\delta_{F_{p-1}}(A \cup B)| \geq p-1$ and $|\delta_{F_{p-1}}(A \cap B)| \geq p-1$. But, by submodularity of $|\delta_{F_{p-1}}(S)|$, we have

$$\begin{aligned} &|\delta_{F_{p-1}}(A)| + |\delta_{F_{p-1}}(B)| \\ &\quad \geq \ |\delta_{F_{p-1}}(A \cap B)| + |\delta_{F_{p-1}}(A \cup B)|. \end{aligned}$$

Hence, we must have $|\delta_{F_{p-1}}(A \cap B)| = |\delta_{F_{p-1}}(A \cup B)| = p - 1$. Thus $h(A \cap B) = h(A \cup B) = 1$. The other case is identical. ■

4 Finding Active Sets

In order to establish a polynomial running time for our approximation algorithm for (IP), we need to show that active sets can be identified and reverse deletes can be performed in polynomial time. Notice that the latter task reduces to the former: an edge e can be deleted if its removal does not create any violated (and hence minimal violated) set. Below, we show that active sets can be found using network flow theory for uncrossable functions that arise from the algorithm of Figure 1.

The manner in which we decompose the problem into phases ensures that minimal violated sets have special structural properties. Consider some phase p of the algorithm, and let F denote the set of currently selected edges. Theorem 4.3 proves that for any active set S there is a choice of vertices $u \in S$ and $v \in \bar{S}$ such that the u-v max flow in the graph $(V, F \cup F_{p-1})$ will help identify S; in particular, S will be the minimal u-v mincut containing u. Once this is established, the implementation will follow from well-known max flow techniques. Theorem 4.3 is based on Lemmas 4.1 and 4.2. In Lemma 4.1 we prove that there is a vertex $u \in S$ such that the u-$\bar{S}$ mincut (i.e., the mincut in the graph obtained by contracting $\bar{S}$ to a single vertex) is S itself and is unique. In Lemma 4.2 we prove that there is a vertex v in $\bar{S}$ such that the S-v max flow has value $p-1$.

Recall that S is violated in the current iteration of the algorithm if $h(S) = 1$ but $\delta_F(S) = \emptyset$. Given the definition of h, this is equivalent to $f(S) \geq p$ and $|\delta_{F_{p-1} \cup F}(S)| = p - 1$.

Lemma 4.1 Let S be a minimal violated set with respect to the set of edges $H = F_{p-1} \cup F$ and the proper function f. Then there exists some $u \in S$, $f(u) \geq p$, such that there is no $T \subset S$, $u \in T$, with $|\delta_H(T)| \leq p - 1$.

Proof: By contradiction. Suppose that for every $u_i \in S$ with $f(u_i) \geq p$ there exists a set $T_i \subset S$ such that $|\delta_H(T_i)| \leq p - 1$. We claim that $f(\bigcup T_i) \leq p - 1$. If this claim is true, then since $f(S) \geq p$, it cannot be the case that $\bigcup T_i = S$. Hence $\bigcup T_i \subset S$, and since $f(S) \geq p$ and $f(\bigcup T_i) \leq p - 1$, it must be the case that $f(S - \bigcup T_i) \geq p$ by Corollary 3.3. But then by maximality there must exist some $u \in (S - \bigcup T_i)$ with $f(u) \geq p$, a contradiction.

Now to prove the claim. By submodularity, for any pair T_i and T_j in the collection of sets $\{T_i\}$, we have $|\delta_H(T_i)| + |\delta_H(T_j)| \geq |\delta_H(T_i - T_j)| + |\delta_H(T_j - T_i)|$. Since both $|\delta_H(T_i)| \leq p - 1$ and $|\delta_H(T_j)| \leq p - 1$, it must be the case that either $|\delta_H(T_i - T_j)| \leq p - 1$ or $|\delta_H(T_j - T_i)| \leq p - 1$. Without loss of generality, suppose the former is true. Then replace T_i in the collection with $T_i - T_j$. Notice that $\bigcup T_i$ remains the same. Continue this process until $\{T_i'\}$ is a collection of pairwise disjoint sets with $\bigcup T_i' = \bigcup T_i$. Since a set is always replaced by a smaller set, this process terminates. Each T_i' has $|\delta_H(T_i')| \leq p - 1$, and since S is a minimal violated set, it must be the case that $f(T_i') \leq p - 1$. Hence by the maximality property of a proper f, $f(\bigcup T_i') \leq p - 1$, which implies that $f(\bigcup T_i) \leq p - 1$. ■

Lemma 4.2 Let S be a violated set with respect to the set of edges $H = F_{p-1} \cup F$ and the proper function f. Then there exists some $u \in S$, $f(u) \geq p$, such that there is no $T \subset S$, $u \in T$, with $|\delta_H(T)| \leq p - 2$.

Proof: As above. In the proof above we used the minimality of S to assert that if $|\delta_H(T_i')| \leq p - 1$, then $f(T_i') \leq p - 1$. In this proof, when $|\delta_H(T_i')| < p - 1$, the fact that H satisfies the requirement for each cut S with $f(S) \leq p - 1$ implies that $f(T_i') < p - 1$. ■

Theorem 4.3 Given any minimal violated set S with respect to $H = F_{p-1} \cup F$ and a proper function f, there exists a pair $u \in S$, $v \notin S$ such that the u-v max flow has value $p - 1$ and S is the minimal u-v mincut containing u.

Proof: Let u be the vertex in S specified by Lemma 4.1. Let v be the vertex in $V - S$ specified by Lemma 4.2 (note that $V - S$ is violated because $f(V - S) = f(S) = p$). Therefore, there is no $T \subset S$ such that $u \in T$ and $|\delta_H(T)| \leq p-1$, and no $T \subset V-S$ such that $v \in T$ and $|\delta_H(T)| \leq p - 2$. Consider the maximum flow from u to v. Because $|\delta_H(S)| = p - 1$, the flow can be no greater than $p-1$. Suppose that the flow is $L \leq p - 1$, and that C is the smallest cut containing u. By hypothesis, it cannot be the case that $C \subset S$. If $C \cap S \neq S$, then because

$$\begin{aligned} L + (p-1) &= |\delta_H(C)| + |\delta_H(S)| \\ &\geq |\delta_H(C \cap S)| + |\delta_H(C \cup S)|, \end{aligned}$$

and because $|\delta_H(C \cap S)| \geq p$, it would follow that $|\delta_H(C \cup S)| < L$. But $C \cup S$ separates u and v; this contradicts the fact that C is a minimum cut. So it must be the case that $S \subseteq C$. Since $V-C \subseteq V-S$ and $v \in V - C$, Lemma 4.2 implies that $|\delta_H(C)| = p - 1$. Since the max flow value is $p - 1$ and $S \subseteq C$, S must be the minimal mincut containing u. ■

Thus we can identify an active set S with one max flow computation, given a good choice of vertices u and v: namely, we compute the u-v max flow, and in the resulting residual graph, select the set of vertices reachable from u. In order to identify all active sets, we first compute the u-v max flow for each ordered pair of vertices u, v, and retain all the minimal mincuts

which are of value $p-1$. We extract from this family the minimal sets under inclusion. We are guaranteed by Theorem 4.3 that the active sets will be among the remaining sets. We then call an oracle for f to determine which of the sets S have value $f(S) \geq p$. These sets will be all the active sets for this iteration of phase p.

The above computation is done at the start of each iteration in a phase. Since a phase has at most n iterations, this requires $O(n^3)$ maximum flow computations per phase. At the end of a phase, reverse delete is performed on the selected edges, which are at most n in number. For testing an edge, we remove it from the graph, and check if there are any minimal violated sets. Therefore, the reverse delete stage also takes $O(n^3)$ maximum flow computations per phase. Hence the running time of our algorithm is essentially the time taken to compute $O(kn^3)$ max flows, since these computations dominate the running time of all other operations. Recently, Gabow, Goemans, and Williamson [4] have shown how to improve the overall running time to $O(k^2n^3)$ time.

We have now completed the description of the algorithm and its implementation. The last two sections are devoted to the proof of the performance guarantee.

5 Proof of Lemma 2.1

In this section, we prove Lemma 2.1. Since the algorithm for (IP_h) maintains the primal complementary slackness conditions, we know that for any edge $e \in F$, $\sum_{S:e\in\delta(S)} y_S = c_e$. Thus the cost of the solution F' is

$$\sum_{e\in F'} c_e = \sum_{e\in F'} \sum_{S:e\in\delta(S)} y_S.$$

We can rewrite the double sum as $\sum_{S\subset V} y_S \cdot |\delta_{F'}(S)|$. To prove the lemma, we will show by induction on the while loop that

$$\sum_{S\subset V} y_S \cdot |\delta_{F'}(S)| \leq \left(2-\frac{2}{\ell}\right) \sum_{S\subset V} h(S) y_S.$$

Since $y_S > 0$ only when $h(S) = 1$, the sum on the right-hand side is effectively $\sum_S y_S$. Certainly the inequality holds before the first iteration of the loop, since initially all $y_S = 0$. Consider the set $\mathcal{C}$ of active sets at the beginning of some iteration of the loop. The left-hand side of the inequality will increase by $\sum_{C\in\mathcal{C}} \epsilon \cdot |\delta_{F'}(C)|$ in this iteration while the increase of the right-hand side will be $(2-\frac{2}{\ell})\epsilon \cdot |\mathcal{C}|$.

The inductive proof will follow from a proof that at any iteration,

$$\sum_{C\in\mathcal{C}} |\delta_{F'}(C)| \leq \left(2-\frac{2}{\ell}\right) |\mathcal{C}|.$$

In other words, we show that the average degree of the active sets with respect to F' is no more than $2-\frac{2}{\ell}$. The proof can be viewed as a charging scheme in which we show that there are many degree one active sets that compensate for high degree active sets. For the remainder of this section, we will concentrate on the active sets $C \in \mathcal{C}$ of some particular iteration of the algorithm, which we will call the current iteration.

Define $H = \bigcup_{C\in\mathcal{C}} \delta_{F'}(C)$; that is, all the edges in F' coming out of active sets. Notice that all these edges must have been added during or after the current iteration.

Lemma 5.1 For any edge $e \in H$ there exists a witness set $S_e \subset V$ such that

1. $h(S_e) = 1$,
2. $\delta_{F'}(S_e) = \{e\}$,
3. For all $C \in \mathcal{C}$ either $C \subseteq S_e$ or $C \cap S_e = \emptyset$.

Proof: Any edge $e \in H$ is also in F', and thus during the reverse delete stage the removal of e causes h to be violated for some S. In other words, there can exist no other $e' \in F'$ that is also in $\delta(S)$. This set S will be the witness set S_e for e, and clearly satisfies (1) and (2). Now let F_b be all the edges added before the current iteration. To show (3), notice that when considering edge e in the reverse delete stage, no edge in F_b had yet been removed. Hence S_e is violated even if all the edges of F_b are included; that is, S_e is violated in the current iteration. Thus (3) follows by the minimality of the active sets C. ■

Consider a collection of sets S_e satisfying the conditions of the preceeding lemma, taken over all the edges e in H. Call such a collection a *witness family*. Any collection of sets is called *laminar* if for any pair of sets A, B in the collection either $A \subseteq B$, $B \subseteq A$, or $A \cap B = \emptyset$.

Lemma 5.2 There exists a laminar witness family.

Proof: By the previous lemma, there exists a witness family. From this collection of sets we can form a laminar collection of sets as follows. We maintain that all sets S in the collection have $h(S) = 1$. If the collection is not laminar, there exists a pair of sets A, B such that $A \not\subseteq B$, $B \not\subseteq A$, and $A \cap B \neq \emptyset$. We say that A and B *cross*. Because $h(A) = h(B) = 1$, either $h(A-B) = h(B-A) = 1$ or $h(A\cup B) = h(A\cap B) = 1$. If the latter is true, we "uncross" A and B by replacing them in the collection with $A \cup B$ and $A \cap B$ (the other case is analogous). This procedure terminates with a laminar collection since whenever two sets are uncrossed, the total number of pairs of sets that cross is reduced. To see this, note that if a set X in the

collection crosses both A and B, then replacing A and B with $A-B$ and $B-A$, or $A \cap B$ and $A \cup B$ cannot increase the total number of sets that X crosses. If X crosses only A, then it cannot cross $B-A$ or $A \cap B$, and so again uncrossing A and B cannot increase the total number of sets that X crosses. Thus uncrossing A and B does not increase the total number of pairs of sets that cross, and in fact decreases the total by at least one, since A no longer crosses B.

We claim that the resulting laminar collection forms a witness family. This claim can be proven by induction on the uncrossing process. Obviously property (3) continues to hold when any two sets are uncrossed. Suppose we have two witness sets S_1 and S_2 corresponding to edges e_1 and e_2 such that S_1 and S_2 cross. Since h is uncrossable, either $h(S_1 \cup S_2) = h(S_1 \cap S_2) = 1$ or $h(S_1 - S_2) = h(S_2 - S_1) = 1$. Without loss of generality, suppose $h(S_1 \cup S_2) = h(S_1 \cap S_2) = 1$. By submodularity, $2 = |\delta_{F'}(S_1)| + |\delta_{F'}(S_2)| \geq |\delta_{F'}(S_1 \cap S_2)| + |\delta_{F'}(S_1 \cup S_2)|$. By the feasibility of F', $|\delta_{F'}(S_1 \cap S_2)| \geq 1$ and $|\delta_{F'}(S_1 \cup S_2)| \geq 1$. Hence it must be the case that if $h(S_1 \cup S_2) = h(S_1 \cap S_2) = 1$, then $|\delta_{F'}(S_1 \cap S_2)| = |\delta_{F'}(S_1 \cup S_2)| = 1$. Therefore either $S_1 \cap S_2$ is a witness set for e_1 and $S_1 \cup S_2$ is a witness set for e_2, or vice versa.■

Let $\mathcal{S}$ be a laminar witness family. Augment the family with the vertex set V. The family can be viewed as defining a tree P with a vertex v_S for each $S \in \mathcal{S}$ and edge (v_S, v_T) if T is the smallest element of $\mathcal{S}$ properly containing S. To each active set $C \in \mathcal{C}$ we correspond the smallest set $S \in \mathcal{S}$ that contains it. We will call a vertex v_S active if S is associated with some active set C.

Lemma 5.3 The tree P has no inactive leaf.

Proof: Only V and the minimal (under inclusion) witness sets can correspond to leaves. Any minimal witness set is a violated set, and thus must contain an active set which corresponds to it. Let S be any maximal witness set. Both S and $V - S$ are violated sets, and thus contain active sets C. Therefore, v_V cannot be simultaneously a leaf and inactive. ■

Lemma 5.4 The degree of an active vertex in P is at least the sum of the $|\delta_{F'}(C)|$ of the $C \in \mathcal{C}$ to which it corresponds.

Proof: Note that the one-to-one mapping between the edges of H and the witness sets implies a one-to-one mapping between the edges of H and the edges of P: each witness set S defines a unique edge (v_S, v_T) of P, where T contains S. Consider any edge $e \in \delta_{F'}(C)$ for some $C \in \mathcal{C}$. Let (v_{S_e}, v_T) be the edge defined by the witness set S_e. Either v_{S_e} or v_T must be the active vertex corresponding to C. By summing over all edges $e \in \delta_{F'}(C)$ for all active sets C corresponding to an active vertex of P, we obtain the lemma. ■

Let P_a denote the set of active vertices in P and let d_v denote the degree of a vertex v. Then, as is also shown in [6],

$$\begin{aligned} \sum_{v \in P_a} d_v &= \sum_{v \in P} d_v - \sum_{v \in P - P_a} d_v \\ &\leq 2(|P| - 1) - 2(|P| - |P_a|) \\ &= 2|P_a| - 2. \end{aligned}$$

This inequality holds since P is a tree with $|P| - 1$ edges, and since each vertex in $P - P_a$ has degree at least 2. The lemma above implies that $\sum_{C \in \mathcal{C}} |\delta_{F'}(C)| \leq \sum_{v \in P_a} d_v$, while clearly $|P_a| \leq |\mathcal{C}|$. Thus

$$\sum_{C \in \mathcal{C}} |\delta_{F'}(C)| \leq 2|\mathcal{C}| - 2.$$

Since $|\mathcal{C}| \leq \ell$, we have that $\sum_{C \in \mathcal{C}} |\delta_{F'}(C)| \leq (2 - \frac{2}{\ell})|\mathcal{C}|$, as desired.

6 Overall Performance Guarantee

Let $A = \{v \in V : f(\{v\}) \geq 1\}$. We are now ready to establish the performance guarantee.

Theorem 6.1 If the function f takes only l distinct values $0 = \rho_0 < \rho_1 < \rho_2 < \ldots < \rho_l$, then the algorithm in Figure 1 produces a feasible set of edges F_k such that

$$\sum_{e \in F_k} c_e \leq \left(2 - \frac{2}{|A|}\right) \sum_{i=1}^{l} \mathcal{H}(\rho_i - \rho_{i-1})\, Z^*_{IP},$$

where $\mathcal{H}$ is the harmonic function $\mathcal{H}(k) = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{k}$ and Z^*_{IP} is the cost of the optimal solution to (IP).

In order to prove Theorem 6.1 from Lemma 2.1, we first show that the dual solution y constructed in phase p by the algorithm can be mapped to a feasible solution to the dual of the linear programming relaxation of (IP). This dual is:

$$\text{Max} \quad \sum_{S \subset V} f(S) y_S - \sum_{e \in E} z_e$$

subject to:

$$(D) \qquad \sum_{S : e \in \delta(S)} y_S \leq c_e + z_e \quad e \in E, \quad (1)$$

$$y_S \geq 0 \qquad \emptyset \neq S \subset V,$$

$$z_e \geq 0 \qquad e \in E.$$

Given the dual variables y constructed by the algorithm in phase p, define $z_e = \sum_{S : e \in \delta(S)} y_S$ for all $e \in F_{p-1}$, and $z_e = 0$ otherwise.

Lemma 6.2 The vector (y, z) is a feasible solution for (D) and $\sum_S y_S = \sum_S g_p(S) y_S - \sum_{e \in E} z_e \leq \sum_S f(S) y_S - \sum_{e \in E} z_e$.

Proof: By the construction of y, for $e \in E_h = E - F_{p-1}$, we know that $\sum_{S: e \in \delta(S)} y_S \leq c_e$. Thus the constraints (1) hold for $e \notin F_{p-1}$. For $e \in F_{p-1}$, the definition of z_e ensures that the constraint (1) holds. This proves that (y, z) is feasible.

By the construction of y, $y_S > 0$ implies that $|\delta_{F_{p-1}}(S)| = p - 1$. By the definition of z_e,

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

Hence $\sum_S g_p(S) y_S - \sum_{e \in E} z_e = p \sum_S y_S - (p-1) \sum_S y_S = \sum_S y_S$, since $y_S > 0$ only when $g_p(S) = p$. The final inequality of the lemma follows since $f(S) \geq g_p(S)$ for all S. ■

We now provide a proof of Theorem 6.1.

Proof: Consider p such that $\rho_{i-1} < p \leq \rho_i$. From Lemmas 2.1 and 6.2, we know that in phase p

$$\begin{aligned} \sum_{e \in F'} c_e &\leq \left(2 - \frac{2}{|A|}\right) \sum_S y_S \\ &= \left(2 - \frac{2}{|A|}\right) \frac{1}{\rho_i - (p-1)} \\ &\qquad \cdot \left(\sum_S \rho_i y_S - \sum_S (p-1) y_S\right) \\ &\leq \left(2 - \frac{2}{|A|}\right) \frac{1}{\rho_i - (p-1)} \left(\sum_S f(S) y_S - \sum_{e \in E} z_e\right) \\ &\leq \left(2 - \frac{2}{|A|}\right) \frac{1}{\rho_i - (p-1)} Z^*_{IP}, \end{aligned}$$

where we have used that $y_S = 0$ if $f(S) < \rho_i$. Summing over all p, we obtain that

$$\begin{aligned} \sum_{e \in F_k} c_e &\leq \left(2 - \frac{2}{|A|}\right) \sum_{i=1}^{l} \sum_{p=\rho_{i-1}+1}^{\rho_i} \frac{1}{\rho_i - (p-1)} Z^*_{IP} \\ &= \left(2 - \frac{2}{|A|}\right) \sum_{i=1}^{l} \mathcal{H}(\rho_i - \rho_{i-1})\, Z^*_{IP}, \end{aligned}$$

proving the desired result. ■

The same bound can be shown to hold even if each edge is allowed some specified number of copies. To prove this, we merely add a distinct primal variable for each copy of the edge.

Theorem 6.1 also applies to the *augmentation* version: Given an initial set of edges F_0, find a minimum-cost set of edges to add to F_0 such that the resulting graph satisfies the proper function f. The only change of the algorithm is in line 1 of Figure 1. The proof of the following theorem is identical to the proof of Theorem 6.1 and is therefore omitted.

Theorem 6.3 If the initial graph satisfies the proper function g_p for some $p \geq 0$ and the values greater or equal to p that f can take are $p = \rho_0 < \rho_1 < \rho_2 < \ldots < \rho_l$, then the algorithm in Figure 1 (with step 1 modified) produces a feasible set of edges F_k such that

$$\sum_{e \in F_k} c_e \leq \left(2 - \frac{2}{|A|}\right) \sum_{i=1}^{l} \mathcal{H}(\rho_i - \rho_{i-1})\, Z^*_{IP},$$

where Z^*_{IP} denotes the optimal value of the augmentation problem.

Theorem 6.1 can be improved by a unit when $\rho_1 = 1$ and $l \geq 2$, as shown in the theorem below. The theorem shows, for example, that the algorithm is a 3-approximation algorithm for the generalized Steiner network problem with $r_{ij} \in \{0, 1, 2\}$.

Theorem 6.4 If the function f takes on values $0 = \rho_0 < \rho_1 = 1 < \rho_2 < \ldots < \rho_l$, $l \geq 2$, then the algorithm in Figure 1 produces a feasible set of edges F_k such that

$$\sum_{e \in F_k} c_e \leq \left(2 - \frac{2}{|A|}\right) \left(\sum_{i=1}^{l} \mathcal{H}(\rho_i - \rho_{i-1}) - \frac{1}{2}\right) Z^*_{IP}.$$

Proof sketch: Let Z_p be the value of the dual solution constructed by the algorithm in phase p; that is, $Z_p = \sum_S p y_S - \sum_{e \in E} z_e$ for the solution (y, z) constructed in the pth phase. Thus Z_p is no greater than the value of the optimal solution to the following linear program:

$$\begin{aligned} &\text{Min} && \sum_{e \in E} c_e x_e \\ &\text{subject to:} \\ (LP_p) \qquad & && \sum_{e \in \delta(S)} x_e \geq p \qquad \forall S : f(S) \geq p \\ & && 0 \leq x_e \leq 1 \qquad e \in E. \end{aligned}$$

In the proof of Theorem 6.1, we used the fact that $Z^*_{IP} \geq Z_p$ for all p. We claim that the following inequality is also true:

$$Z^*_{IP} \geq Z_1 + \frac{1}{2} Z_2.$$

To see this, consider the optimal solution x^* to (IP) and let H be the corresponding set of edges. This optimal network consists of several maximal 2-connected

blocks interconnected by bridges. Let C denote the set of bridges of H. We decompose x^* into the sum of two vectors α and β where:

$$\alpha_e = \begin{cases} \frac{x_e^*}{2} & \text{if } e \notin C \\ 0 & \text{otherwise} \end{cases}$$

and

$$\beta_e = \begin{cases} \frac{x_e^*}{2} & \text{if } e \notin C \\ x_e^* & \text{otherwise.} \end{cases}$$

Clearly, $x^* = \alpha + \beta$. We assert that, by definition of C, β is a feasible solution to (LP_1), and 2α is a feasible solution to (LP_2). The first assertion is easy to see; we omit the proof of the second due to space constraints. Given these assertions,

$$Z_{IP}^* = \sum_{e \in E} c_e x_e^* = \sum_{e \in E} c_e \beta_e + \sum_{e \in E} c_e \alpha_e \geq Z_1 + \frac{1}{2} Z_2.$$

From the proofs of Lemma 6.2 and Theorem 6.1, we know that

$$\sum_{e \in F_k} c_e \leq \left(2 - \frac{2}{|A|}\right) \left\{ Z_1 + \sum_{i=2}^{l} \mathcal{H}(\rho_i - \rho_{i-1})\, Z_i \right\}.$$

One can show that the above inequality and the observations that $Z_{IP}^* \geq Z_p$ for all p and $Z_{IP}^* \geq Z_1 + \frac{1}{2} Z_2$ imply the desired result. ■

7 Concluding Remarks

Extending our algorithm to handle non-uncrossable functions remains a challenging open problem. The key feature of uncrossable functions is that there exists an optimal dual solution y which is laminar: that is, the sets S such that $y_S > 0$ form a laminar family. This property characterizes uncrossable functions. Handling all non-uncrossable functions is ruled out by the fact that there exist instances corresponding to non-uncrossable $\{0,1\}$ functions whose ratio between Z_{IP}^* and the optimal dual value is larger than any constant. Another open problem is to improve the approximation ratio for proper functions. In particular, we do not know whether the analysis of our algorithm is tight: currently, we only have tight examples for $k \leq 2$.

A larger open issue is to explore further the power of the primal-dual approach for obtaining approximation algorithms for other combinatorial optimization problems. This appears to be quite promising.

Acknowledgements

We wish to thank Naveen Garg and Neal Young for valuable discussions. The first author would like to thank David Johnson and AT&T for allowing him to spend the summer at Bell Labs.

References

[1] A. Agrawal, P. Klein and R. Ravi, "When trees collide: An approximation algorithm for the generalized Steiner problem in networks", *Proc. 23rd ACM Symp. on Theory of Computing*, 134–144 (1991).

[2] P. Berman and V. Ramaiyer, "Improved approximations for the Steiner tree problem", *Proc. 3rd ACM-SIAM Symp. on Discrete Algorithms*, 325–334 (1992).

[3] G.N. Frederickson and J. Ja'Ja', "Approximation algorithms for several graph augmentation problems", *SIAM J. Comput.* 10, 270–283 (1981).

[4] H.N. Gabow, M.X. Goemans, and D.P. Williamson, "An Efficient Approximation Algorithm for the Survivable Network Design Problem," to appear in *Proc. Third Conference on Integer Programming and Combinatorial Optimization.*

[5] M.X. Goemans and D.J. Bertsimas, "Survivable networks, linear programming relaxations and the parsimonious property", to appear in *Math. Programming.*

[6] M.X. Goemans and D.P. Williamson, "A general approximation technique for constrained forest problems", *Proc. 3rd ACM-SIAM Symp. on Discrete Algorithms*, 307–316 (1992).

[7] M. Grötschel, C.L. Monma and M. Stoer, "Design of survivable networks", to appear in the *Handbook in Operations Research and Management Science*, Eds: Michael Ball, Thomas Magnanti, Clyde Monma, and George Nemhauser (1992).

[8] S. Khuller and U. Vishkin, "Biconnectivity approximations and graph carvings", Technical Report UMIACS-TR-91-132, Univ. of Maryland (September 1991). Also appears in *Proc. 24th ACM Symp. on Theory of Computing*, 759–770 (1992).

[9] C.H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity.* Englewood Cliffs, NJ: Prentice-Hall (1982).

[10] R. Ravi and P. Klein, "When cycles collapse: A general approximation technique for constrained two-connectivity problems", Technical Report CS-92-30, Brown University (June 1992). Also to appear in *Proc. Third Conference on Integer Programming and Combinatorial Optimization.*

[11] H. Saran, V. Vazirani, and N. Young, "A primal-dual approach to approximation algorithms for network Steiner problems", *Proc. of Indo-US workshop on Cooperative Research in Computer Science*, Bangalore, India, 166–168 (1992).

[12] A.Z. Zelikovsky, "The 11/6 approximation algorithm for the Steiner problem on networks", to appear in *Information and Computation.*