Algorithmica (1997) 18: 21–43

Algorithmica

An Approximation Algorithm for Minimum-Cost Vertex-Connectivity Problems

R. Ravi[1] and D. P. Williamson[2]

Abstract. We present an approximation algorithm for solving graph problems in which a low-cost set of edges must be selected that has certain vertex-connectivity properties. In the survivable network design problem, a value r_{ij} for each pair of vertices i and j is given, and a minimum-cost set of edges such that there are r_{ij} vertex-disjoint paths between vertices i and j must be found. In the case for which $r_{ij} \in \{0, 1, 2\}$ for all i, j, we can find a solution of cost no more than three times the optimal cost in polynomial time. In the case in which $r_{ij} = k$ for all i, j, we can find a solution of cost no more than $2\mathcal{H}(k)$ times optimal, where $\mathcal{H}(n) = 1 + \frac{1}{2} + \cdots + \frac{1}{n}$. No approximation algorithms were previously known for these problems. Our algorithms rely on a primal–dual approach which has recently led to approximation algorithms for many edge-connectivity problems.

Key Words. Approximation algorithm, Vertex connectivity, Survivable network design, Primal–dual method.

1. Introduction. Let $G = (V, E)$ be an undirected graph with nonnegative costs $c_e \geq 0$ on all edges $e \in E$. In the *survivable network design problem*, a nonnegative integer r_{ij} for each pair of vertices i, j is given, and a minimum-cost set of edges $E' \subseteq E$ must be found such that for every i, j pair there are at least r_{ij} vertex-disjoint paths between i and j in the graph (V, E'). The *minimum-cost k-vertex-connectivity problem* is a special case of the survivable network design problem in which $r_{ij} = k$ for all pairs of vertices i, j.

The survivable network design problem has received a good deal of attention in the literature recently, as it can be used to model the design of low-cost telephone networks that can "survive" certain types of edge and node failures. An edge cost c_e denotes the cost of laying a fiber-optic cable between the endpoints of edge e, and a value r_{ij} denotes the number of edge and node failures that must occur in the network before i and j are completely disconnected. In practice, the values of r_{ij} tend to be quite low, usually no more than 2 for all vertex pairs, since failures are assumed to be isolated accidents, such as fires at nodes [15], [20]. We call the problem in which $r_{ij} \in \{0, 1, 2\}$ the $\{0, 1, 2\}$-survivable network design problem. The survivable network design problem is known to be NP-hard even in the case of the minimum-cost 2-vertex-connectivity problem [2], even if the edge costs are either 1 or 2. Because of this, many heuristics have been devised

[1] Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburgh, PA 15213, USA. ravi@andrew.cmu.edu. This research was supported by NSF Grant CCR-91-03937 and a DIMACS postdoctoral fellowship, and was conducted in part while the author was visiting MIT.

[2] IBM TJ Watson Research Center, Room 33-219, P.O. Box 218, Yorktown Heights, NY 10598, USA. dpw@watson.ibm.com. This research was supported by an NSF Graduate Fellowship and an NSF Postdoctoral Fellowship, and was conducted in part while the author was a graduate student at MIT and in part while a postdoc at Cornell.

Received October 27, 1994; revised August 14, 1995. Communciated by M. X. Goemans.

to find solutions to the survivable network design problem; see Chapter 3 of [20] for a survey. In particular, Monma and Shallcross [15] devised local improvement heuristics for a special case of the $\{0, 1, 2\}$-survivable network design problem arising from Bellcore network design problems; these heuristics were used in a Bellcore software package. Grötschel *et al.* [5] implemented a branch-and-cut algorithm for the same problem.

A difficulty with such heuristics for the survivable network design problem, however, is that the solution produced may not have cost guaranteed to be close to the cost of the optimal solution. For this reason, we consider *approximation algorithms* for special cases of the survivable network design problem. An α-approximation algorithm is a polynomial-time algorithm that produces a solution of cost no more than α times the value of an optimal solution. We give the first known approximation algorithm for the most practical variant of the survivable network design problem in which $r_{ij} \in \{0, 1, 2\}$ for all pairs of vertices i, j. The algorithm produces solutions of value no more than three times the optimal value. Furthermore, we give the first known approximation algorithm for the minimum-cost k-vertex-connectivity problem. Our algorithm produces solutions of value no more than $2\mathcal{H}(k)$ times optimal, where $\mathcal{H}(n) = 1 + \frac{1}{2} + \cdots + \frac{1}{n}$. Our techniques also provide approximation algorithms for the problem of augmenting an l-vertex-connected graph to a k-vertex-connected graph using edges of minimum cost. We give solutions for this problem of value no more than $2\mathcal{H}(k - l)$ times optimal.

Our work flows out of a recent line of research on designing approximation algorithms for edge-connectivity problems [1], [4], [3], [12], [24]. This research has led to a $2\mathcal{H}(\max_{i,j} r_{ij})$-approximation algorithm for the survivable network design problem in which there must be r_{ij} *edge*-disjoint paths between vertices i and j. In fact, very general types of connectivity problems can be approximated in which, for each subset of vertices S, there must be at least $f(S)$ edges selected from $\delta(S)$, where $\delta(S) = \{(u, v) \in E : u \in S, v \notin S\}$ and $f(S)$ is a function $f: 2^V \to \mathbf{N}$ of a certain form. We follow the approach of these algorithms, and their proofs, particularly those given in [24] and [3]. The algorithms in these papers break down the problem into a number of *phases*. In each phase we specify certain vertex sets S that must be augmented; that is, we must select an additional edge from $\delta(S)$ of each specified set S. This augmentation problem is formulated as an integer programming problem, and the problem is solved by using a variant of the primal–dual method. For a more detailed presentation of the algorithm and an overview of this line of research, see [22].

Implementations of the edge-connectivity algorithms have shown that they work well in practice [23], [14], coming within a few percent of optimal. We expect that the same will also be true of our new vertex-connectivity algorithms. We further observe that the solutions given by our algorithm can be used as a good starting point for improvement heuristics like those of Monma and Shallcross [15] or as a bounding routine within a branch-and-bound scheme.

Very few optimal or approximation algorithms were known for solving minimum-cost vertex-connectivity problems prior to our work. Khuller and Thurimella [11] give a 3-approximation algorithm for the minimum-cost 2-vertex-connectivity problem. Khuller [10] shows a $2(1 + 1/n)$-approximation algorithm for the same problem in an n-vertex graph. After the appearance of an extended abstract of our paper [18], Khuller and Raghavachari [10] gave a $(2 + 2(k - 1)/n)$-approximation algorithm for the minimum-cost k-vertex-connectivity problem for graphs with edge costs that obey the triangle

inequality. Our results do not require the triangle inequality assumption. Most of the known results for vertex-connectivity problems are for the restricted case in which the graph G is complete and all edge costs are identical. In this case, Harary [6] has shown how to find an optimal solution for the k-vertex-connectivity problem. In this setting, Jordán [9] has shown how to augment any k-vertex-connected graph to a $(k+1)$-vertex-connected graph using at most $k-2$ more edges than necessary. Optimal algorithms are known to augment any starting graph to a 2-vertex-connected graph [2] or to a 3-vertex-connected graph [21], [8]. Hsu [7] has shown how to augment optimally any 3-vertex-connected graph to a 4-vertex-connected graph.

The rest of the paper is structured as follows. Section 3 describes the approximation algorithms by showing how they can be reduced to an augmentation algorithm. Section 4 proves that the algorithms provide near-optimal solutions. Section 5 tells how the algorithms can be implemented in polynomial time, and we conclude in Section 6 with a few remarks. We first begin with some preliminary definitions and concepts in Section 2.

2. Notation and Basic Definitions. Given a set of edges I and a set of vertices S, we define $\Gamma_I(S)$ to be the "vertex neighborhood" of S with respect to I; that is, $\Gamma_I(S) = \{v \in V - S : (u, v) \in I \text{ for some } u \in S\}$. The "vertex complement" of S, $\zeta_I(S)$, is defined to be $V - S - \Gamma_I(S)$. Occasionally we drop the subscript I when it is clear from the context.

We need some facts about Γ and ζ. First, $\Gamma_I(S)$ is submodular for any edge set I. That is, for any edge set I and any two sets of vertices A and B, $|\Gamma_I(A)| + |\Gamma_I(B)| \geq |\Gamma_I(A \cup B)| + |\Gamma_I(A \cap B)|$. We also observe that $\Gamma_I(A \cap B) - A - B \subseteq \Gamma_I(A) \cap \Gamma_I(B)$. Finally, for any vertex set A, $A \subseteq \zeta_I(\zeta_I(A))$.

For a set of edges I and a set of vertices S, the *coboundary* of S is denoted $\delta_I(S)$ and defined to be $\{(u, v) \in I | u \in S, v \notin S\}$. We define $\delta_I(S : T)$ to be the set of edges $\{(u, v) \in I | u \in S, v \in T\}$.

Given a set of edges $I \subseteq E$ of a graph $G = (V, E)$, a set $C \subseteq V$ is a *cutset* of I if there are fewer connected components in (V, I) than in the graph induced by removing the vertices in C and adjacent edges from I. If $C = \{v\}$ is a cutset for some vertex v, the vertex is called a *cutvertex*. A cutset C *separates* two vertices s and t if s and t are in the same connected component of (V, I), and removing C causes s and t to be in different connected components.

We use the following theorem of Menger [13].

THEOREM 2.1 (Menger). *Let s and t be two nonadjacent vertices in a connected graph G. Then there exist at least k vertex-disjoint paths between s and t if and only if there is no cutset of size $k-1$ or less separating s and t.*

Menger's theorem has the following simple corollary.

COROLLARY 2.2. *Given a connected graph G with at least $k+1$ vertices, the graph is not k-vertex-connected if and only if there exists a cutset of size $k-1$ or less separating a pair of nonadjacent vertices.*

Notice that any graph that contains a k-vertex-connected subgraph must contain at least $k + 1$ vertices; we assume that the graphs given as input to our k-vertex-connectivity algorithm have at least $k + 1$ vertices. Therefore, if a connected graph on $k + 1$ vertices has no nonadjacent vertices with a cutset of size $k - 1$ or smaller separating them, then every pair of vertices will have k-vertex-disjoint paths between them.

3. The Algorithms. In this section we give the basic high-level structure of the algorithms. We show how the $\{0, 1, 2\}$-survivable network design problem and the k-vertex-connectivity problem can both be reduced to an augmentation problem. Given a set of edges F to augment, we specify sets of vertices S such that we must select at least one additional edge from $\delta(S : \zeta_F(S))$ for each S. Intuitively speaking, this will increase the size of the vertex neighborhood of S by at least one. We then give an algorithm AUGMENT which is able to find a low-cost solution to this problem under certain conditions.

3.1. *The Algorithm for the Minimum-Cost k-Vertex-Connectivity Problem.* We begin by giving the high-level structure of the algorithm for the minimum-cost k-vertex-connectivity problem, which we call APPROX-k-VERTEX-CONN. The algorithm consists of k *phases*, and each phase adds edges to the current solution, starting initially with the empty set of edges at the beginning of phase 1. The idea is that at the end of phase p, our set of edges should form a p-vertex-connected graph on the set of vertices of the input graph. We denote our set of edges at the end of phase p as F_p. In phase p we must augment the $(p - 1)$-vertex-connected edge set F_{p-1} to the p-vertex-connected F_p.

In order to perform this augmentation, we call the augmenting subroutine AUGMENT. AUGMENT takes as input the edge set F_{p-1} and a function h: $2^V \to \{0, 1\}$, and returns a set of edges $F' \subseteq E - F_{p-1}$ such that if $h(S) = 1$, then $|\delta_{F'}(S : \zeta_{F_{p-1}}(S))| \geq 1$. The idea is that adding F' to F_{p-1} will increase the size of the vertex neighborhood of S by at least 1 for each S such that $h(S) = 1$. The function h has exponential size in the number of vertices, so we would never be able to write the function down when calling AUGMENT, but we will see in Section 5 that we will be able to answer AUGMENT's queries about h in polynomial time.

In phase p of APPROX-k-VERTEX-CONN, we set $h(S) = 1$ exactly when the vertex neighbors of S are a size $p - 1$ cutset of F_{p-1}, and S is the smaller of the two halves of the graph induced by removing the vertex neighborhood of S, $\Gamma_{F_{p-1}}(S)$. Thus $h(S) = 1$ in phase p iff $|\Gamma_{F_{p-1}}(S)| = p - 1$, and $0 < |S| \leq \lfloor (n - (p - 1))/2 \rfloor$. We claim then that $F_p \leftarrow F_{p-1} \cup F'$ is p-vertex-connected. Because F_{p-1} is $(p - 1)$-vertex connected, there can be no cutsets of size $p - 2$ or smaller, and by the definition of h and AUGMENT, we increase the size of the vertex neighborhood of any set S which has a size $p - 1$ cutset in the edge set F_{p-1}. Thus F_p is p-vertex-connected.

The overall algorithm is given in Figure 1.

3.2. *The Algorithm for the* $\{0, 1, 2\}$*-Survivable Network Design Problem.* In this section we present the high-level algorithm for solving the survivable network design problem when $r_{ij} \in \{0, 1, 2\}$. The algorithm here will have two phases. In the first phase we find a solution for the network design problem with requirements $r'_{ij} = \max(r_{ij} - 1, 0)$.

```
APPROX-k-VERTEX-CONN (V, E, c, k)
1   F_0 ← ∅
2   for p ← 1 to k
3     Comment: Begin phase p
4     h(S) ← { 1  if |Γ_{F_{p-1}}(S)| = p − 1 and 0 < |S| ≤ ⌊(n−(p−1))/2⌋
             { 0  otherwise
5     F' ← AUGMENT(V, E, F_{p-1}, c, h)
6     F_p ← F' ∪ F_{p-1}
7   return F_k
```

Fig. 1. The algorithm for k-vertex-connectivity.

In the second phase we augment this solution to find a feasible solution for the original problem. Essentially the first phase finds a set of edges to connect all pairs of vertices i, j for which $r_{ij} = 2$, and the second phase finds a second vertex-disjoint path for these pairs of vertices, as well as edges to connect all pairs i, j for which $r_{ij} = 1$.

Notice that when finding a set of edges to solve the problem of the first phase, there is no difference between edge-connectivity and vertex-connectivity. Hence we can use an approximation algorithm for the edge-connectivity survivable network design problem in which $r'_{ij} \in \{0, 1\}$. As was mentioned in the Introduction, such algorithms are already known; the first such algorithm for this problem is due to Agrawal *et al.* [1]. For the sake of our analysis, we use the algorithm of Goemans and Williamson [4]. Let F_1 denote the set of edges returned by this algorithm.

To find an augmenting set of edges in the second phase, we once again use the subroutine AUGMENT. We call AUGMENT on a modified graph $G' = (V', E')$. For every edge $(i, j) \in F_1$ such that $r_{ij} = 2$, we create a new vertex u that subdivides the edge (i, j). That is, for each such edge (i, j), we add u to V, remove (i, j) from E, and add edges (i, u) and (u, j). Our reason for doing this is so that i and j are no longer adjacent in the modified graph, and so that we will be able to apply Menger's theorem.[3] Note that there is a solution F'_1 in G' corresponding to F_1 in G.

To use AUGMENT, we set $h(S) = 1$ exactly when there exist $i \in S$, $j \in \zeta_{F'_1}(S)$ (in the modified graph G') such that $|\Gamma_{F'_1}(S)| < r_{ij}$. This can happen in one of two ways. If $r_{ij} = 2$, then i and j need to be 2-vertex-connected but are separated by a cutvertex. If $r_{ij} = 1$, then i and j need to be connected, but are in different connected components. As before, AUGMENT will return a set of edges $F' \subseteq E' - F'_1$ such that $|\delta_{F'}(S : \zeta_{F'_1}(S))| \geq 1$ whenever $h(S) = 1$. By the definition of h and F'_1, for any two i, j such that $r_{ij} = 2$, the set of edges $F'_1 \cup F'$ will contain no cutvertex separating i and j in the modified graph. Similarly, for any two i, j such that $r_{ij} = 1$, there will exist a path in $F'_1 \cup F'$ from i to j in the modified graph. Notice that since there are no edges in $E' - F'_1$ involving the new vertices of the modified graph, the set of edges F' is a subset of edges of the original graph. Thus $F_2 \leftarrow F_1 \cup F'$ is a feasible solution to the original problem in the original graph: the nonexistence of a cutvertex separating any i, j with $r_{ij} = 2$ in the modified

[3] We do not need to subdivide edges in APPROX-k-VERTEX-CONN because of Corollary 2.2.

APPROX-0,1,2-SNDP (V, E, c, r)

1 $r'_{ij} \leftarrow \max(r_{ij} - 1, 0)$ for all i, j

2 $F_1 \leftarrow$EDGE-SNDP(V, E, c, r')

3 Modify V, E and F_1 by creating for every edge $(i, j) \in F_1$ such that $r_{ij} = 2$ a new vertex u that subdivides the edge (i, j). Call the resulting sets V', E' and F'_1 respectively.

4 $h(S) \leftarrow \begin{cases} 1 & \text{if } \exists i \in S, j \in \zeta_{F'_1}(S) \text{ such that } |\Gamma_{F'_1}(S)| < r_{ij} \\ 0 & \text{otherwise} \end{cases}$

5 $F' \leftarrow$AUGMENT(V', E', F'_1, c', h)

6 $F_2 \leftarrow F' \cup F_1$

7 **return** F_2

Fig. 2. The algorithm for the {0,1,2}-survivable network design problem.

graph implies that there are two vertex-disjoint paths between i and j in the edge set F_2, and similarly that i and j are connected in F_2 if $r_{ij} = 1$.

The overall algorithm is given in Figure 2.

3.3. *The* AUGMENT *Algorithm.* The augmentation algorithm AUGMENT is given in Figure 3. It is adapted from the algorithm of Williamson *et al.* [24] used in the edge-connectivity survivable network design algorithms. Given a graph (V, E), a set of edges I, and an input function h meeting three conditions, the algorithm produces a low-cost set of edges $F' \subseteq E_h \equiv E - I$ such that $|\delta_{F'}(S : \zeta_I(S))| \geq h(S)$ for all nontrivial subsets S. Before discussing the conditions on h, we first introduce some definitions.

DEFINITION 3.1. A vertex set S is *violated* with respect to a set of edges F' if $h(S) = 1$ but $\delta_{F'}(S : \zeta_I(S)) = \emptyset$.

Thus F' is a feasible solution for the augmentation problem if there are no violated sets with respect to F'.

DEFINITION 3.2. A vertex set S is *active* with respect to a set of edges F' if it is violated and minimal with respect to inclusion.

DEFINITION 3.3. Two sets of vertices A, B are *crossing* (or A *crosses* B) if $A \cap B \neq \emptyset$ and neither $A \subseteq B$ nor $B \subseteq A$.

We can now state the first two conditions on h.

CONDITION 1. For any edge set $F \subseteq E_h$, it must be the case that no violated set with respect to F crosses any active set with respect to F.

CONDITION 2. For any edge set $F \subseteq E_h$, the active sets of h with respect to F can be computed in polynomial time.

```
AUGMENT (V, E, I, c, h)
1   i ← 0
2   d(e) ← 0 for all e ∈ E_h ≡ E − I
3   F ← ∅
4   C ← active sets with respect to F
5   a(e) ← { 2  if e ∈ δ(C_1 : ζ_I(C_1)) ∩ δ(C_2 : ζ_I(C_2)) for C_1, C_2 ∈ C, C_1 ≠ C_2
           { 0  if e ∉ δ(C : ζ_I(C)) for any C ∈ C
           { 1  otherwise
6   while |C| > 0
7     i ← i + 1
8     Comment: Begin iteration i
9     Find edge e_i = (u, v), a(e_i) ≠ 0, that minimizes ε = (c_e − d(e)) / a(e)
10    For all e ∈ E_h
11      d(e) ← d(e) + a(e) · ε
12    Comment: Implicitly set y_C ← y_C + ε for all C ∈ C
13    F ← F ∪ {e_i}
14    Update C
15    Update a(e)
16    Comment: End iteration i
17  Comment: Edge deletion stage
18  F' ← F
19  for j ← i downto 1
20    If F' − {e_j} is feasible
21      F' ← F' − {e_j}
22  return F'
```

Fig. 3. The augmenting algorithm AUGMENT.

Notice that the first condition implies that all active sets are disjoint. The third condition on h is a more technical one which we will introduce in the analysis of the algorithm. We prove that the first condition holds for the functions h from the algorithms at the end of the section. We show that the second condition holds for our algorithms in Section 5. The three conditions are summarized at the end of Section 4.3.

The algorithm AUGMENT works in two stages. In the first stage the algorithm starts with an empty set of edges F and goes through a sequence of *iterations*. In each iteration an edge is added to F. The first stage terminates when F is a feasible solution (that is, there are no violated sets). To be more specific, we denote the collection of active sets with respect to the current set of edges F by $\mathcal{C}$. In each iteration the algorithm selects an edge from $\delta_{E_h}(C : \zeta_I(C))$ for some active set $C \in \mathcal{C}$, and adds this edge to F. Clearly when there are no longer any active sets, the set of edges F is a feasible solution. The second stage deletes redundant edges from F to obtain F'. To do this, we consider all the edges of F in the reverse of the order in which they were added to F. If removing the edge from F does not affect the feasibility of the remaining edge set, we discard it.

We use duality to guide the addition of edges to F, in order to ensure that we find a low-cost solution. In effect, we would like to solve the following integer program:

$$(AUG) \qquad \text{Min} \sum_{e \in E_h} c_e x_e$$

$$\text{subject to} \quad \sum_{e \in \delta_{E_h}(S:\zeta_I(S))} x_e \geq h(S), \qquad \emptyset \neq S \subset V,$$

$$x_e \in \{0, 1\}, \qquad e \in E_h.$$

Consider the dual of the linear programming relaxation of (AUG):

$$(D) \qquad \text{Max} \sum_{S} h(S) y_S$$

$$(1) \qquad \text{subject to} \quad \sum_{S: e \in \delta(S:\zeta_I(S))} y_S \leq c_e, \qquad e \in E_h,$$

$$y_S \geq 0.$$

Our algorithm will maintain a feasible solution for (D). At the beginning of the algorithm we set $y_S = 0$ for all $S \subset V$. In each iteration we increase the dual variables y_C uniformly for all currently active sets C until one of the packing constraints (1) becomes tight; that is, for some $e \in E_h$,

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

In particular, the constraint must become tight for some edge $e \in \delta_{E_h}(C : \zeta_I(C))$ for some $C \in \mathcal{C}$. We choose to add this edge e to F in this iteration.

In the description of AUGMENT given in Figure 3, we only use the dual variables y_S implicitly. Instead, we maintain variables $a(e)$ and $d(e)$ such that in each iteration $a(e)$ indicates the number of active sets C such that $e \in \delta_{E_h}(C : \zeta_I(C))$ (note that this can be either 0, 1, or 2) and

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

Thus in each iteration we can increase each y_C for the active sets C by ε as long as

$$d(e) + a(e) \cdot \varepsilon \leq c_e.$$

Therefore, the largest ε can be is the minimum of $(c_e - d(e))/a(e)$ taken over all $e \in \delta(C : \zeta_I(C))$ over all active sets C. An edge e that attains this minimum will be added to F. Notice that the implicit dual solution remains feasible for (D). It is initially feasible since, for any edge e, $\sum_{S: e \in \delta(S:\zeta_I(S))} y_S = 0 \leq c_e$. Once an edge e is selected, $\sum_{S: e \in \delta(S:\zeta_I(S))} y_S$ does not increase because no S such that $e \in \delta(S : \zeta_I(S))$ will be violated.

We now turn to showing that active sets do not cross any violated sets for the functions h used by the algorithms of the preceding sections. We begin with functions h of APPROX-k-VERTEX-CONN.

LEMMA 3.4. *Let h be a function from phase p of* APPROX-k-VERTEX-CONN, *let F_{p-1} be the edges found in the first $p-1$ phases of* APPROX-k-VERTEX-CONN, *and let $F \subseteq E - F_{p-1}$. If A and B are crossing violated sets with respect to the edge set F, then $A \cap B$ is a violated set and either $A \cup B$ or $\zeta_{F \cup F_{p-1}}(A \cup B)$ is a violated set.*

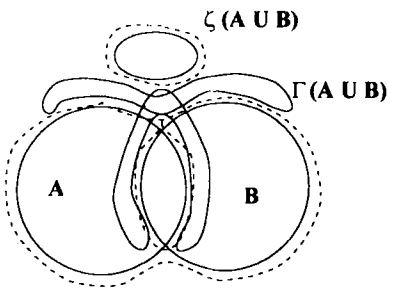

Fig. 4. Cases in the proof of Lemma 3.4. Note that both $\zeta(A \cup B) \neq \emptyset$ and $\zeta(A \cap B) \neq \emptyset$. The candidate uncrossed sets are drawn in thin broken lines. $A \cap B$ and the smaller of $A \cup B$ and $\zeta(A \cup B)$ are the uncrossed sets.

PROOF. Recall that in this case a set S is violated iff $\delta_F(S : \zeta_{F_{p-1}}(S)) = \emptyset$, and $h(S) = 1$, which is equivalent to $\zeta_{F \cup F_{p-1}}(S) \neq \emptyset$, $|\Gamma_{F \cup F_{p-1}}(S)| = p - 1$, $0 < |S| \leq \lfloor (n - (p - 1))/2 \rfloor$. We use this equivalent definition in the proof. For the remainder of the proof we drop the subscript $F \cup F_{p-1}$ from the δ, ζ, and Γ functions.

The strategy of the proof is to show that both $|\Gamma(A \cap B)| \geq p - 1$ and $|\Gamma(A \cup B)| \geq p - 1$. Hence by submodularity and the fact that $|\Gamma(A)| = |\Gamma(B)| = p - 1$, we have that $|\Gamma(A \cap B)| = |\Gamma(A \cup B)| = p - 1$, immediately implying that $A \cap B$ is violated and, a little less immediately (as we will show), that the smaller of $A \cup B$ and $\zeta(A \cup B)$ is violated. In order to show that $|\Gamma(A \cap B)| \geq p - 1$ and $|\Gamma(A \cup B)| \geq p - 1$, we show that $\zeta(A \cap B)$ and $\zeta(A \cup B)$ are nonempty. Then the fact that F_{p-1} is $(p - 1)$-vertex-connected implies that $|\Gamma(A \cap B)| \geq p - 1$ in order for there to be $p - 1$ vertex-disjoint paths between the vertices in $A \cap B$ and $\zeta(A \cap B)$. Similarly, $\zeta(A \cup B) \neq \emptyset$ implies $|\Gamma(A \cup B)| \geq p - 1$.

Now to begin the proof. Because A and B are violated and cross, we know that $|A \cup B| \leq n - (p - 1) - 1$. We claim that $\zeta(A \cap B) \neq \emptyset$. Using the relation from Section 2 that $\Gamma(A \cap B) - A - B \subseteq \Gamma(A) \cap \Gamma(B)$, we see that $|(A \cap B) \cup \Gamma(A \cap B)| \leq |A \cup B \cup (\Gamma(A) \cap \Gamma(B))| \leq n - 1$, implying that $\zeta(A \cap B) \neq \emptyset$. By the logic of the preceding paragraph, it follows that $|\Gamma(A \cap B)| \geq p - 1$. By the submodularity of $|\Gamma(S)|$, it follows that $|\Gamma(A \cup B)| \leq p - 1$. Then $|A \cup B \cup \Gamma(A \cup B)| \leq n - 1$, or $\zeta(A \cup B) \neq \emptyset$. Then $|\Gamma(A \cup B)| \geq p - 1$. Thus it follows by submodularity that $|\Gamma(A \cup B)| = |\Gamma(A \cap B)| = p - 1$. Thus $A \cap B$ is certainly violated. Also, we claim that the smaller of $A \cup B$ or $\zeta(A \cup B)$ must be violated. Certainly the smaller of them must have size no greater than $\lfloor (n - (p - 1))/2 \rfloor$. If $A \cup B$ is smaller, then we are done, as we have already shown that $\zeta(A \cup B) \neq \emptyset$ and $|\Gamma(A \cup B)| = p - 1$. If $\zeta(A \cup B)$ is smaller, then, since $A \cup B \subseteq \zeta(\zeta(A \cup B))$, we know $\zeta(\zeta(A \cup B)) \neq \emptyset$. In addition, $|\Gamma(\zeta(A \cup B))| = p - 1$ since $\Gamma(\zeta(A \cup B)) \subseteq \Gamma(A \cup B)$, and since the $(p - 1)$-vertex-connectivity of F_{p-1} implies that $|\Gamma(\zeta(A \cup B))| \geq p - 1$. □

THEOREM 3.5. *Let h be a function from phase p of* APPROX-k-VERTEX-CONN, *and let F be any set of edges. The active sets with respect to the edge set F do not cross any violated set.*

PROOF. Suppose there is an active set A that crosses a violated set B. Then by Lemma 3.4, $A \cap B$ is also a violated set, with $A \cap B \subset A$, contradicting the minimality of A. □

Now we consider the case of the survivable network design problem. Let h be the function from APPROX-0,1,2-SNDP, let F_1 be the set of edges found in Step 2 of APPROX-0,1,2-SNDP, and let $F \subset E - F_1$. In this case a set S is violated iff (1) there exist $i \in S$, $j \in \zeta_{F_1}(S)$ such that $|\Gamma_{F_1 \cup F}(S)| = 1$ and $r_{ij} = 2$; or (2) there exist $i \in S$, $j \in \zeta_{F_1}(S)$ such that $|\Gamma_{F_1 \cup F}(S)| = 0$ and $r_{ij} = 1$.

LEMMA 3.6. *Let h be the function from* APPROX-0,1,2-SNDP, *let F_1 be the set of edges found in Step 2 of* APPROX-0,1,2-SNDP, *and let $F \subseteq E - F_1$. If A and B are crossing violated sets, then either $A \cap B$ and $A \cup B$ are violated, $A - (B \cup \Gamma(B))$ and $B - (A \cup \Gamma(A))$ are violated, or $A - B$ and $B - A$ are violated.*

PROOF. Notice that if neither A nor B is a violated set of type (1), then the set A (or B) is violated iff $g(A) \equiv \max(h(A) - |\delta_{F_1 \cup F}(A)|, 0) = 1$. Goemans *et al.* [3] have shown that this function is uncrossable; i.e., if $g(A) = g(B) = 1$, then either $g(A - B) = g(B - A) = 1$ or $g(A \cup B) = g(A \cap B) = 1$. Thus if A and B are violated sets of type (2), then so are either $A - B$ and $B - A$, or $A \cap B$ and $A \cup B$.

We must show what happens if one of the two sets is of type (1). Suppose B is a violated set of type (1). Let i and j be the pair of vertices separated by A and let i' and j' be the pair separated by B, where $i \in A$ and $i' \in B$. The different cases are depicted in Figure 5.

Case 1. A is also of type (1). Notice that the single vertex neighbor of A can be in either $B - A$ or $V - (A \cup B)$. Similarly the vertex neighbor of B can be in either $A - B$ or $V - (A \cup B)$. We consider the various cases. First suppose that both A and B have their vertex neighbor in $V - (A \cup B)$. This case has two subcases. First, if there is an edge from $A \cap B$ to $V - (A \cup B)$, then there can be no edges from $A - B$ to $A \cap B$ or $B - A$ to $A \cap B$, since then A and/or B would have more than one vertex neighbor (Figure 5(a)). Thus in order for F_1 to be feasible for the requirements r'_{ij}, it must be the case that $i, i' \in A \cap B$ and $j, j' \in V - (A \cup B)$. Since $\Gamma(A) = \Gamma(B) = \Gamma(A \cup B) = \Gamma(A \cap B)$ in this subcase, it follows that $A \cap B$ and $A \cup B$ are violated. The other subcase is that in which there is no edge from $A \cap B$ to $V - (A \cup B)$. Then it follows that there is an edge with endpoints in $A - B$ and $V - (A \cup B)$, and another with endpoints in $B - A$ and $V - (A \cup B)$. It also follows that $i \in A - B$, $i' \in B - A$, and $j, j' \in V - (A \cup B)$ (Figure 5(b)). Since then $\Gamma(A) = \Gamma(A - B)$ and $\Gamma(B) = \Gamma(B - A)$, it follows that $A - B$ and $B - A$ are violated.

Next, we suppose that the vertex neighbor of A is in $B - A$ and the vertex neighbor of B is in $V - (A \cup B)$. It follows that $i \in A \cap B$ (since otherwise B has more than one vertex neighbor or F_1 is not feasible) and $j' \in V - (A \cup B)$ (Figure 5(c)). Then $\Gamma(A) = \Gamma(A \cap B)$, $\Gamma(B) = \Gamma(A \cup B)$, and $A \cap B$ and $A \cup B$ are violated. The case in which the vertex neighbor of B is in $A - B$ and the vertex neighbor of A is in $V - (A \cup B)$ is parallel to this one.

The only remaining case is when the vertex neighbor of A is in $B - A$ and the vertex neighbor of B is in $A - B$. Then it must be the case that $j \in B - (A \cup \Gamma(A))$

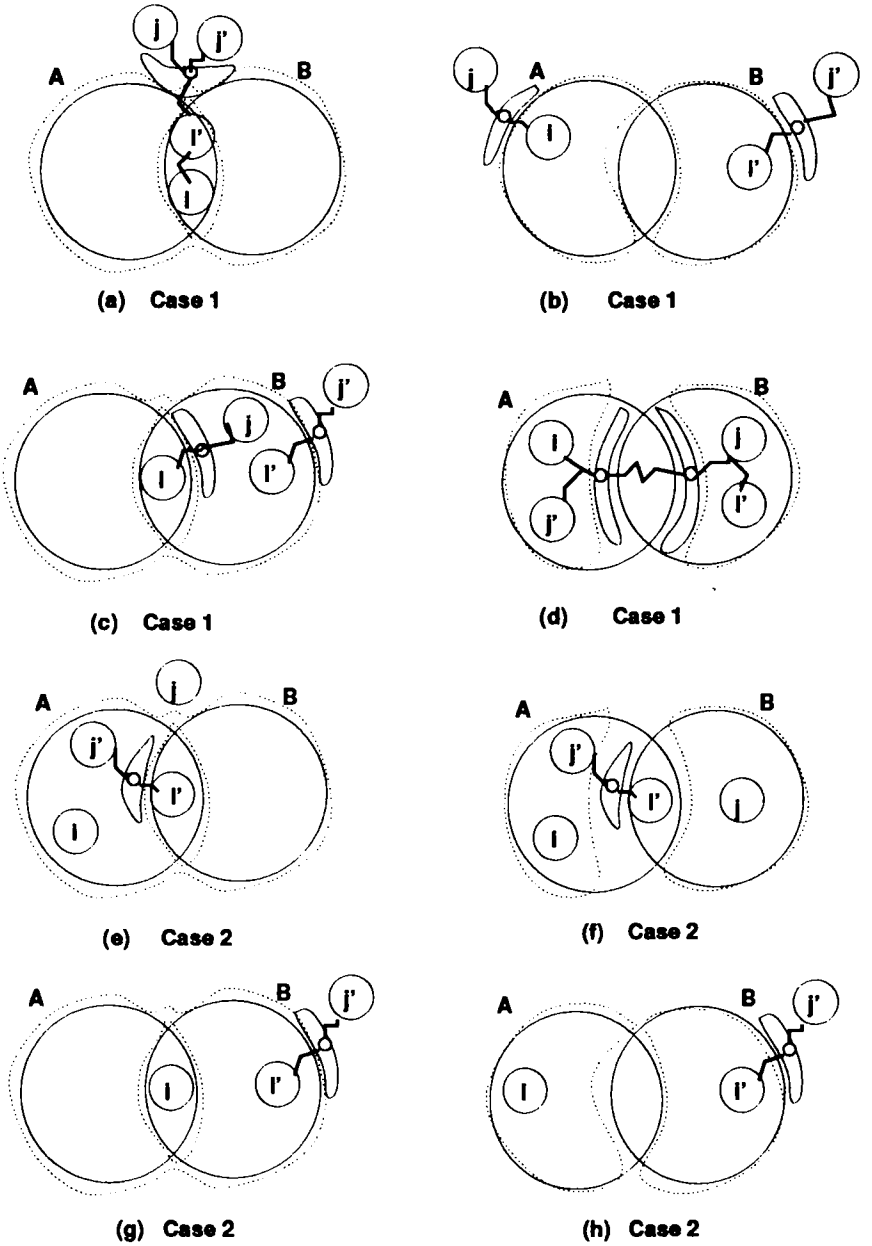

Fig. 5. Cases in the proof of Lemma 3.6. The uncrossed sets are drawn in thin broken lines while the paths in F_1 are depicted by thick lines.

and $j' \in A - (B \cup \Gamma(B))$ (Figure 5(d)). Since $\Gamma(A) = \Gamma(B - (A \cup \Gamma(A)))$ and $\Gamma(B) = \Gamma(A - (B \cup \Gamma(B)))$, then $A - (B \cup \Gamma(B))$ and $B - (A \cup \Gamma(A))$ are violated.

Case 2. A is of type (2). Here the vertex neighbor of B can be in either $A - B$ or $V - (A \cup B)$. If the vertex neighbor of B is in $A - B$, then there are two cases. By reasoning similar to that above, it must be the case that $i' \in A \cap B$. If $j \in V - (A \cup B)$, then $\Gamma(A) = \Gamma(A \cup B)$, $\Gamma(B) = \Gamma(A \cap B)$, and $A \cap B$ and $A \cup B$ are violated (Figure 5(e)). If $j \in B - A$, then $\Gamma(A) = \Gamma(B - (A \cup \Gamma(A))) = \emptyset$, $\Gamma(B) = \Gamma(A - (B \cup \Gamma(B)))$, and $A - (B \cup \Gamma(B))$ and $B - (A \cup \Gamma(A))$ are violated (Figure 5(f)). Now if the vertex neighbor of B is in $V - (A \cup B)$, there are two possibilities. If $i \in A \cap B$, then $\Gamma(A \cap B) = \Gamma(A) = \emptyset$, $\Gamma(A \cup B) = \Gamma(B)$, and $A \cap B$ and $A \cup B$ are violated (Figure 5(g)). If $i \in A - B$, then $\Gamma(A - B) = \Gamma(A) = \emptyset$, $\Gamma(B - A) = \Gamma(B)$, and thus $A - B$ and $B - A$ are violated (Figure 5(h)). □

The lemma implies the following theorem.

THEOREM 3.7. *Let h be the function from* APPROX-0,1,2-SNDP, *let F_1 be the set of edges found in Step* 2 *of* APPROX-0,1,2-SNDP, *and let $F \subseteq E - F_1$. The active sets with respect to the edge set F do not cross any violated set.*

PROOF. As in Theorem 3.5. □

4. Analysis of the Algorithms. We now turn to the proofs that the algorithms of the previous section provide solutions whose cost is close to the optimal cost. At the heart of our proofs is a theorem about the solutions produced by AUGMENT.

THEOREM 4.1. *For the functions h used in* APPROX-k-VERTEX-CONN *and* APPROX-0,1,2-SNDP, AUGMENT *produces a feasible set of edges F' and a feasible dual solution y to (D) such that*

$$\sum_{e \in F'} c_e \leq 2 \sum_S h(S) \cdot y_S.$$

We first show how this theorem implies that APPROX-k-VERTEX-CONN is a $2\mathcal{H}(k)$-approximation algorithm and that APPROX-0,1,2-SNDP is a 3-approximation algorithm, before proving the theorem itself.

4.1. *Analysis of* APPROX-k-VERTEX-CONN. We now prove the following theorem, where Z^*_{kVC} is the value of an optimal solution to the given instance of the minimum-cost k-vertex-connectivity problem.

THEOREM 4.2. *The algorithm* APPROX-k-VERTEX-CONN *produces a k-vertex-connected set of edges F_k such that*

$$\sum_{e \in F_k} c_e \leq 2\mathcal{H}(k) Z^*_{kVC}.$$

PROOF. Fix an optimal solution to the instance of the k-vertex-connectivity problem, and let $x_e^* = 1$ if edge e is in the solution, and $x_e^* = 0$ otherwise. The key observation needed for the theorem is that in phase p of the algorithm, $(1/(k-p+1))x^*$ is a feasible solution to the linear programming relaxation of (AUG) for the function h used in phase p. In phase p, any S for which $h(S) = 1$ has $|\Gamma_{F_{p-1}}(S)| = p - 1$. Thus it must be the case that x^* has at least $k - (p-1)$ edges between S and $\zeta_{F_{p-1}}(S)$, otherwise x^* is not k-vertex-connected. In other words,

$$\sum_{e \in \delta_{E - F_{p-1}}(S : \zeta_{F_{p-1}}(S))} x_e^* \geq [k - (p-1)] \cdot h(S),$$

so that $(1/(k-p+1))x^*$ is a feasible solution to the linear programming relaxation of (AUG).

Therefore, for any feasible solution y to (D) in phase p, by weak duality, it must be the case that

$$\sum_S h(S) \cdot y_S \leq \frac{1}{k-p+1} Z_{kVC}^*.$$

By Theorem 4.1, the cost of edges added in phase p is no more than $2/(k-p+1)Z_{kVC}^*$. Summing over all phases we have that

$$\sum_{e \in F_k} c_e \leq \sum_{p=1}^{k} \frac{2}{k-p+1} Z_{kVC}^* = 2\mathcal{H}(k) Z_{kVC}^*. \qquad \square$$

COROLLARY 4.3. *A modification of* APPROX-k-VERTEX-CONN *gives a* $2\mathcal{H}(k-l)$*-approximation algorithm for the problem of adding a minimum-cost set of edges to an* l*-vertex-connected subgraph to make it* k*-vertex-connected,* $l < k$.

PROOF. We modify APPROX-k-VERTEX-CONN by changing lines 1 and 2 of Figure 1. In line 1 we set F_l to the edges of the l-vertex-connected subgraph of $G = (V, E)$. In line 2 we iterate p from $l+1$ to k. By a proof similar to that of Theorem 4.2, the solution returned by the AUGMENT routine in the phase p has cost at most $2/((k-l)-p+1)$ times that of an optimal augmentation. Summing over the phases gives the claimed performance guarantee. $\square$

4.2. *Analysis of* APPROX-0,1,2-SNDP. Let Z_{SN}^* denote the cost of an optimal solution to the given instance of the $\{0, 1, 2\}$-survivable network design problem. We prove the following theorem in a manner similar to the previous theorem.

THEOREM 4.4. *The algorithm* APPROX-0,1,2-SNDP *produces a set of edges* F_2 *such that*

$$\sum_{e \in F_2} c_e \leq 3Z_{SN}^*.$$

PROOF. As in the previous theorem, let x^* denote an optimal solution to the problem. To prove the result, we show two things: first, that the cost of the edges returned by the

Goemans–Williamson algorithm is no more than Z^*_{SN}, and, second, that the cost of the augmenting edges is no more than $2Z^*_{SN}$. These two results imply the theorem.

The Goemans–Williamson algorithm finds a feasible solution to the following integer program:

$$(IP) \qquad \text{Min} \sum_{e \in E} c_e x_e$$

$$\text{subject to} \quad \sum_{e \in \delta(S)} x_e \geq \max_{i \in S, j \notin S} r'_{ij}, \qquad \emptyset \neq S \subset V,$$

$$x_e \in \{0, 1\}, \qquad e \in E.$$

As in the algorithm AUGMENT, the construction of the (IP) solution is guided by the construction of a solution to the dual of the linear programming relaxation. Goemans and Williamson [4] show that the cost of the (IP) solution constructed is no more than twice the cost of the solution to the dual of the linear programming relaxation. Thus $\sum_{e \in F_1} c_e$ is no more than twice the value of the optimal solution to the linear programming relaxation. Note that $\frac{1}{2}x^*$ is a feasible solution for the linear programming relaxation of (IP). From this it follows that $\sum_{e \in F_1} c_e \leq Z^*_{SN}$.

We now show that $\sum_{e \in F'} c_e \leq 2Z^*_{SN}$ for the function h used by APPROX-0,1,2-SNDP. Notice that given an optimal solution x^*, there is a corresponding solution x' to the same problem for the modified graph G' of the same cost. Observe that x' is a feasible solution for (AUG) for the function h. Then, by Theorem 4.1, we are done. □

4.3. *Analysis of* AUGMENT. We wish to prove Theorem 4.1, and show that

$$\sum_{e \in F'} c_e \leq 2 \sum_S h(S) \cdot y_S.$$

The proof is similar to the proof in Section 5 of [24]. Recall that I is a set of edges given as input to AUGMENT, and F' is the set of edges output by the algorithm. Notice that, for every edge $e \in F'$, $c_e = \sum_{S : e \in \delta(S : \zeta_I(S))} y_S$, and $y_S > 0$ only if $h(S) = 1$, so that the inequality can be rewritten as

$$\sum_{e \in F'} \sum_{S : e \in \delta(S : \zeta_I(S))} y_S \leq 2 \sum_S y_S.$$

Rewriting again gives

$$\sum_S y_S \cdot |\delta_{F'}(S : \zeta_I(S))| \leq 2 \sum_S y_S.$$

We can prove this statement by induction over the iterations of the algorithm. Initially $y_S = 0$ for all $S \subseteq V$, and the statement is true. At each iteration, if $\mathcal{C}$ is the current collection of active sets, then the left-hand side of the inequality increases by

$$\varepsilon \cdot \sum_{C \subset \mathcal{C}} |\delta_{F'}(C : \zeta_I(C))|,$$

while the right-hand side increases by $2\varepsilon|\mathcal{C}|$. Hence if we can prove that in any iteration,

$$\sum_{C \in \mathcal{C}} |\delta_{F'}(C : \zeta_I(C))| \leq 2|\mathcal{C}|,$$

then the theorem will be proved.

We now fix some particular iteration of the algorithm, which we call the "current iteration." Let F denote the set of edges chosen in iterations up to (but not including) the current iteration. For convenience, we implicitly assume that all δ, Γ, and ζ functions have a subscript of $I \cup F$ unless otherwise noted. Define $Y = \bigcup_{C \in \mathcal{C}} \delta_{F'}(C : \zeta(C))$. Notice that all the edges in Y must have been added during or after the current iteration.

LEMMA 4.5. *For each edge $e \in Y$ there exists a witness set $S_e \subset V$ such that:*

(1) $\delta_{F'}(S_e : \zeta(S_e)) = \{e\}$.
(2) *S_e is violated in the current iteration.*
(3) *For each $C \in \mathcal{C}$ either $C \subseteq S_e$ or $C \cap S_e = \emptyset$.*

PROOF. Any edge $e \in Y$ is also in F', and thus during the edge deletion stage the removal of e causes there to exist some violated set; call this set S. In other words, there can exist no other $e' \in F'$ that is also in $\delta_{F'}(S : \zeta(S))$. This set S will be the witness set for e, and clearly satisfies (1). Now let F be all the edges added before the current iteration. To show (2) and (3), notice that when considering edge e in the edge deletion stage, no edge in F had yet been removed. Hence S_e is violated even if all the edges of F are included; that is, S_e is violated in the current iteration. Property (3) follows by the fact that no active set crosses any violated set S_e. □

Consider a collection of sets S_e satisfying the conditions of the preceding lemma, taken over all the edges e in Y. We call such a collection a *witness family*. A family of sets is called *laminar* if no two sets of the family are crossing.

We can now state the final condition we require on h (the three conditions are summarized at the end of this section). The additional technical restriction we require on the function h in order for Theorem 4.1 to hold is the following.

CONDITION 3. There exists a laminar witness family.

At the end of the section we show that this condition holds for the functions h used by APPROX-k-VERTEX-CONN and APPROX-0,1,2-SNDP. The remaining proof of the inequality is essentially identical to the proof of Williamson *et al.*, but we include it here for the sake of completeness. 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 H 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 associate the smallest set $S \in \mathcal{S}$ that contains it. We color the vertices of the tree H: a vertex v_S is colored red if S is associated with some active set C and colored blue otherwise. Let $\mathcal{L}(v_S)$ be the collection of active sets associated with a red vertex v_S.

LEMMA 4.6. *The tree H has at most one blue 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. Thus only V can correspond to a blue leaf. □

LEMMA 4.7. *For any red vertex v_S in H, the degree of v_S is at least $\sum_{C \in \mathcal{L}(v_S)} |\delta_{F'}(C : \zeta(C))|$.*

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

Let H_r denote the set of red vertices in H and let d_v denote the degree of a vertex v in H. Then

$$\sum_{v \in H_r} d_v = \sum_{v \in H} d_v - \sum_{v \in H - H_r} d_v \leq 2(|H| - 1) - 2(|H| - |H_r| - 1) - 1 = 2|H_r| - 1.$$

This inequality holds since H is a tree with $|H| - 1$ edges, and since all vertices of $H - H_r$ except for possibly one have degree at least 2. The lemma above implies that $\sum_{C \in \mathcal{C}} |\delta_{F'}(C : \zeta(C))| \leq \sum_{v \in H_r} d_v$, while clearly $|H_r| \leq |\mathcal{C}|$. Thus

$$\sum_{C \in \mathcal{C}} |\delta_{F'}(C : \zeta(C))| \leq 2|\mathcal{C}|,$$

as desired.

We now recall the three necessary conditions on h.

CONDITIONS ON h.

1. For any edge set $F \subseteq E_h$, no violated set with respect to F crosses any active set with respect to F.
2. For any edge set $F \subseteq E_h$, the active sets with respect to F can be computed in polynomial time.
3. In any iteration of AUGMENT, there exists a laminar witness family.

We can now state the following corollary to Theorem 4.1.

COROLLARY 4.8. *For any function h such that the first and third conditions are obeyed,* AUGMENT *produces a set of edges F' and a dual feasible solution y such that*

$$\sum_{e \in F'} c_e \leq 2 \sum_S h(S) \cdot y_S.$$

In Section 5 we also show that if the second condition is obeyed, then AUGMENT runs in polynomial time. Together with Corollary 4.8, this implies that AUGMENT is a 2-approximation algorithm for the integer program (*AUG*) for any function h that obeys the three conditions.

4.4. *Laminar Witness Families for* APPROX-k-VERTEX-CONN. We now turn to proving Condition 3 for the function h used by APPROX-k-VERTEX-CONN; in the next subsection we prove it for the function h of APPROX-0,1,2-SNDP. In both cases we show that there exists a laminar witness family by "uncrossing" pairs of sets using Lemmas 3.4 and 3.6.

For the algorithm APPROX-k-VERTEX-CONN, we first need the following lemmas.

LEMMA 4.9. *If A is a violated set with respect to the function h in phase p of* APPROX-k-VERTEX-CONN, *then $\zeta(\zeta(A)) = A$.*

PROOF. It is not hard to see that $A \subseteq \zeta(\zeta(A))$. Suppose there exists a vertex $v \in \zeta(\zeta(A)) - A$. Then it must be the case that $v \in \Gamma(A)$, $\zeta(A \cup \{v\}) = \zeta(A)$, and $\Gamma(A \cup \{v\}) = \Gamma(A) - \{v\}$. Since A is violated, $\zeta(A) \neq \emptyset$ and $|\Gamma(A)| = p - 1$. However, then $\zeta(A \cup \{v\}) \neq \emptyset$ and $|\Gamma(A \cup \{v\})| < p - 1$, which contradicts the feasibility of the edge set F_{p-1}. □

LEMMA 4.10. *If A and B are crossing violated sets, then $\Gamma(A) \cap \Gamma(B) \subseteq \Gamma(A \cap B)$ and $\Gamma(A) \cap B \subseteq \Gamma(A \cap B)$.*

PROOF. In general $\Gamma(A \cap B) \subseteq \Gamma(A) \cup \Gamma(B)$ and $\Gamma(A \cup B) \subseteq \Gamma(A) \cup \Gamma(B)$. If A and B are crossing violated sets, then we know that

$$|\Gamma(A)| + |\Gamma(B)| = |\Gamma(A \cap B)| + |\Gamma(A \cup B)|,$$

so that any vertex appearing k times in the sets on the right-hand side of the equation ($k = 0, 1, 2$) must appear exactly k times in the sets on the left-hand side, and vice versa. This immediately implies $\Gamma(A) \cap \Gamma(B) \subseteq \Gamma(A \cap B)$. Also, since no vertex in $\Gamma(A) \cap B$ can be in $\Gamma(A \cup B)$, then $\Gamma(A) \cap B \subseteq \Gamma(A \cap B)$. □

LEMMA 4.11. *Let $\mathcal{S}$ be a collection of violated sets with respect to the function h in phase p of* APPROX-k-VERTEX-CONN. *Then there exists a laminar family of violated sets formed by successively replacing a crossing pair of sets A and B with an appropriate choice of $A \cap B$ and $A \cup B$, or $A \cap B$ and $\zeta(A \cup B)$.*

PROOF. We use a potential function

$$\Phi(\mathcal{S}) = \sum_{S \in \mathcal{S}} (|S|^2 + |\zeta(S)|^2)$$

to prove the lemma. If A and B cross and are both violated sets, then by Lemma 3.4 either $A \cap B$ and $A \cup B$, or $A \cap B$ and $\zeta(A \cup B)$ are both violated. Let $\mathcal{S}'$ be the collection of sets formed by replacing A and B with the pair of violated sets. We will show that

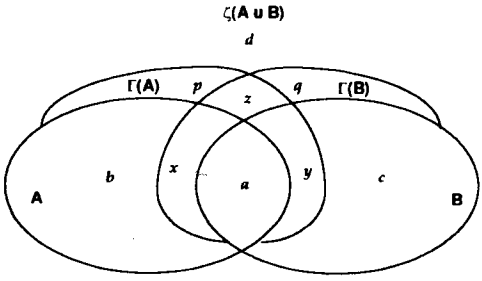

Fig. 6. Two crossing sets A and B.

$\Phi(\mathcal{S}') - \Phi(\mathcal{S}) > 0$. Since uncrossing pairs of sets does not increase the number of sets in the collection, Φ can never grow larger than $2|\mathcal{S}| \cdot n^2$. Thus the uncrossing process must terminate with a laminar family.

Let

$$a = |A \cap B|, \quad b = |A - \Gamma(B) - B|, \quad c = |B - \Gamma(A) - A|,$$
$$x = |A \cap \Gamma(B)|, \quad y = |B \cap \Gamma(A)|, \quad z = |\Gamma(A) \cap \Gamma(B)|,$$
$$p = |\Gamma(A) - \Gamma(B) - B|, \quad q = |\Gamma(B) - \Gamma(A) - A|, \quad d = |\zeta(A \cup B)|$$

(see Figure 6). The initial contribution of A to $\Phi(\mathcal{S})$ is $(a+x+b)^2+(c+q+d)^2$, and the initial contribution of B is $(a+y+c)^2+(b+p+d)^2$. After uncrossing, the contribution of $A \cap B$ is at least $a^2+(b+p+d+q+c)^2$ and the contribution of $A \cup B$ (or $\zeta(A \cup B)$, which we treat symmetrically by Lemma 4.9) is at least $d^2+(a+b+c+x+y)^2$. Since all other sets in $\mathcal{S}'$ stay the same, $\Phi(\mathcal{S}') - \Phi(\mathcal{S})$ is at least the difference between these two quantities, which, by algebraic manipulation, is $2((b+p)(c+q)+(x+b)(y+c))$. Since A and B are crossing, $|A-B| > 0$ and $|B-A| > 0$, which implies that $x+b > 0$ and $y+c > 0$. Thus $\Phi(\mathcal{S}') - \Phi(\mathcal{S}) > 0$. □

We can now prove Condition 3 for the function h used by APPROX-k-VERTEX-CONN.

LEMMA 4.12. *For the function h given in phase p of* APPROX-k-VERTEX-CONN, *there exists a laminar witness family.*

PROOF. By Lemma 4.5, 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 are violated. If the collection is not laminar, there exists a pair of sets A, B that cross. We "uncross" A and B by replacing them in the collection with either $A \cup B$ and $A \cap B$ or with $\zeta(A \cup B)$ and $A \cap B$. By Lemma 3.4, we know that at least one of these two uncrossings yields two violated sets. This procedure terminates with a laminar collection by Lemma 4.11.

We claim that the resulting laminar collection forms a witness family. This claim can be proven by induction on the uncrossing process. Recall that each set in the witness family must obey three properties:

(1) $\delta_{F'}(S_e : \zeta(S_e)) = \{e\}$.

(2) S_e is violated in the current iteration.
(3) For each $C \in \mathcal{C}$ either $C \subseteq S_e$ or $C \cap S_e = \emptyset$.

Obviously property (2) holds. Property (3) continues to hold because the uncrossed sets are violated sets for the current iteration, and must either contain or be disjoint from the minimal violated sets. Now we must prove (1). Suppose we have two crossing witness sets S_1 and S_2 corresponding to edges e_1 and e_2, and, without loss of generality (by Lemma 4.9) suppose we uncross them into $S_1 \cap S_2$ and $S_1 \cup S_2$. We claim that

$$|\delta_{F'}(S_1 : \zeta(S_1))| + |\delta_{F'}(S_2 : \zeta(S_2))| \geq |\delta_{F'}(S_1 \cup S_2 : \zeta(S_1 \cup S_2))| + |\delta_{F'}(S_1 \cap S_2 : \zeta(S_1 \cap S_2))|.$$

We want to show that each edge counted on the right-hand side is accounted for by the same edge on the left-hand side. If an edge is in $\delta_{F'}(S_1 \cup S_2 : \zeta(S_1 \cup S_2))$, then certainly it is in either $\delta_{F'}(S_1 : \zeta(S_1))$ or $\delta_{F'}(S_2 : \zeta(S_2))$; it is possibly in both, and certainly in both if it is also in $\delta_{F'}(S_1 \cap S_2 : \zeta(S_1 \cap S_2))$. Consider an edge in $\delta_{F'}(S_1 \cap S_2 : \zeta(S_1 \cap S_2))$. If its endpoint in $\zeta(S_1 \cap S_2)$ is in $V - (S_1 \cup S_2)$, then the edge is in either $\delta_{F'}(S_1 : \zeta(S_1))$ or $\delta_{F'}(S_2 : \zeta(S_2))$: the endpoint cannot be in both $\Gamma(S_1)$ and $\Gamma(S_2)$, since we know $\Gamma(S_1) \cap \Gamma(S_2) \subseteq \Gamma(S_1 \cap S_2)$ by Lemma 4.10. Suppose that the endpoint in $\zeta(S_1 \cap S_2)$ is in $S_1 - S_2$. By Lemma 4.10, $\Gamma(S_2) \cap S_1 \subseteq \Gamma(S_1 \cap S_2)$, implying that the endpoint cannot be in $\Gamma(S_2)$ and must be in $\zeta(S_2)$. Thus the edge is also in $\delta_{F'}(S_2 : \zeta(S_2))$. A similar argument holds if the endpoint is in $S_2 - S_1$.

Because F' covers all violated sets, we know that $|\delta_{F'}(S_1 \cup S_2 : \zeta(S_1 \cup S_2))| \geq 1$ and $|\delta_{F'}(S_1 \cap S_2 : \zeta(S_1 \cap S_2))| \geq 1$, and so it must be the case that $|\delta_{F'}(S_1 \cup S_2 : \zeta(S_1 \cup S_2))| = |\delta_{F'}(S_1 \cap S_2 : \zeta(S_1 \cap S_2))| = 1$. Then either $e_1 \in \delta_{F'}(S_1 \cup S_2 : \zeta(S_1 \cup S_2))$ and $e_2 \in \delta_{F'}(S_1 \cap S_2 : \zeta(S_1 \cap S_2))$, or vice versa. □

4.5. *Laminar Witness Families for* APPROX-0,1,2-SNDP. In this section we prove that Condition 3 is obeyed by the function h used by APPROX-0,1,2-SNDP.

LEMMA 4.13. *For the function h given by* APPROX-0,1,2-SNDP, *there exists a laminar witness family.*

PROOF. As before, we know a witness family exists, and our strategy is to show that we can form a laminar witness family by uncrossing any crossing pairs of sets. By Lemma 3.6, whenever two violated sets A and B cross, then either $A \cap B$ and $A \cup B$ are violated, or $A - (B \cup \Gamma(B))$ and $B - (A \cup \Gamma(A))$ are violated, or $A - B$ and $B - A$ are violated. We replace any pair of crossing witness sets with the appropriate pair of violated, noncrossing sets. This process terminates, since the number of pairs of crossing sets decreases. If any set X crosses both A and B, uncrossing A and B does not increase the number of sets X crosses. If X crosses A and either contains B or is disjoint from B, then it cannot cross $A \cap B$, $B - A$, or $B - (A \cup \Gamma(A))$. If X crosses A and is contained in B, then it cannot cross $A \cup B$, $A - B$, or $A - (B \cup \Gamma(B))$. Thus the total number of pairs of crossing sets does not increase, and must decrease by at least one since A and B no longer cross.

As before, we prove that the resulting laminar family of sets is a witness family by

induction on the uncrossing process. Each set in the witness family must obey three properties:

(1) $\delta_{F'}(S_e : \zeta(S_e)) = \{e\}$.
(2) S_e is violated in the current iteration.
(3) For each $C \in \mathcal{C}$ either $C \subseteq S_e$ or $C \cap S_e = \emptyset$.

As before, properties (2) and (3) follow straightforwardly, and we must show that (1) holds. Suppose we have two crossing witness sets S_1 and S_2 corresponding to edges e_1 and e_2. The proof of Lemma 3.6 shows that if S_1 and S_2 are uncrossed into $S_1 - S_2$ and $S_2 - S_1$, then $\Gamma(S_1) = \Gamma(S_1 - S_2), \Gamma(S_2) = \Gamma(S_2 - S_1)$, and $\Gamma(S_1), \Gamma(S_2) \subset V - (S_1 \cup S_2)$ (Figure 5(b) and (h)). Therefore, by a simple counting argument,

$$|\delta_{F'}(S_1 : \zeta(S_1))| + |\delta_{F'}(S_2 : \zeta(S_2))| \geq |\delta_{F'}(S_1 - S_2 : \zeta(S_1 - S_2))| + |\delta_{F'}(S_2 - S_1 : \zeta(S_2 - S_1))|,$$

and property (1) holds in the same way as shown in Lemma 4.12. The proof of Lemma 3.6 shows that if S_1 and S_2 are uncrossed into $S_1 - (S_2 \cup \Gamma(S_2))$ and $S_2 - (S_1 \cup \Gamma(S_1))$, then $\Gamma(S_1) = \Gamma(S_2 - (S_1 \cup \Gamma(S_1))) \subset S_2 - S_1$ and $\Gamma(S_2) = \Gamma(S_1 - (S_2 \cup \Gamma(S_2))) \subset S_1 - S_2$ (Figure 5(d) and (f)). In this case we claim a counting argument shows that

$$\begin{aligned}|\delta_{F'}(S_1 : \zeta(S_1))| + |\delta_{F'}(S_2 : \zeta(S_2))| &\geq |\delta_{F'}(S_1 - (S_2 \cup \Gamma(S_2)) : \zeta(S_1 - (S_2 \cup \Gamma(S_2))))| \\ &\quad + |\delta_{F'}(S_2 - (S_1 \cup \Gamma(S_1)) : \zeta(S_2 - (S_1 \cup \Gamma(S_1))))|.\end{aligned}$$

The only tricky case is when an edge in $\delta_{F'}(S_1 - (S_2 \cup \Gamma(S_2)) : \zeta(S_1 - (S_2 \cup \Gamma(S_2))))$ has an endpoint of $\Gamma(S_1)$, since then the edge is not in $\delta_{F'}(S_1 : \zeta(S_1))$. However, in this case the edge must be in $\delta_{F'}(S_2 : \zeta(S_2))$. So we can again infer that property (1) holds. Likewise, if S_1 and S_2 are uncrossed into $S_1 \cap S_2$ and $S_1 \cup S_2$, then a similar counting argument shows that

$$|\delta_{F'}(S_1 : \zeta(S_1))| + |\delta_{F'}(S_2 : \zeta(S_2))| \geq |\delta_{F'}(S_1 \cup S_2 : \zeta(S_1 \cup S_2))| + |\delta_{F'}(S_1 \cap S_2 : \zeta(S_1 \cap S_2))|.$$

Again, property (1) holds as was argued in Lemma 4.12. □

5. Implementation. We now turn to the problem of implementing the algorithm AUGMENT. We must show how to find active sets for the algorithms APPROX-k-VERTEX-CONN and APPROX-0,1,2-SNDP, how to select the edge minimizing ε in each iteration, and how to remove edges.

As in the case of the edge-connectivity approximation algorithms, active sets can be found using network flow theory, although it is slightly more complicated in this case. Suppose that there is a active set S (i.e., a minimal violated set) with respect to the edge set $I \cup F$. In both the case of APPROX-k-VERTEX-CONN and APPROX-0,1,2-SNDP this is because $|\Gamma_{I \cup F}(S)| < r_{st}$ for some $s \in S$, $t \in \zeta_{I \cup F}(S)$. We can determine S as follows. Construct a directed graph $G' = (V', E')$ from the graph $(V, I \cup F)$ by making

two copies v', v'' for each $v \in V$, adding directed edges (u'', v') and (v'', u') of infinite capacity for each $(u, v) \in I \cup F$, and adding an edge (v', v'') of unit capacity for each $v \in V$. It is known that the value of a maximum s''–t' flow in G' corresponds to the number of vertex-disjoint paths between s and t in G [16, p. 458]. Furthermore, the minimal mincut in G' will correspond to S. The minimal mincut is given by the vertices reachable from s in the residual graph of the flow. In particular, the vertices v in S are those such that both v' and v'' are on the source side of the directed cut, the vertices v in $\zeta(S)$ are those for which both v' and v'' are on the sink side of the cut, and the vertices in $\Gamma(S)$ are those for which v' is on the source side and v'' is on the sink side.

Thus a straightforward way of finding active sets is to calculate an s–t maximum flow for all pairs of vertices $s, t \in V$, find all the minimal mincuts, check if the mincut value is less than r_{st} to see if the set is violated, then extract all the minimal violated sets from this collection. There will be $O(n^2)$ candidate sets, and we claim that the minimal violated sets can be extracted from the candidates in $O(n^3)$ time. For each vertex, we calculate the set of smallest cardinality containing it, and the minimal violated sets will be all the sets of smallest cardinality.

We can cut down the total time used by keeping track of the residual graphs for each network flow problem. Whenever the algorithm AUGMENT adds an edge to F, we add the edge to each s–t residual graph, and see if it makes any more vertices reachable from s. Given the active sets from the previous iteration, we can then extract the new active sets in $O(n^2)$ time. Let $r_{\max} = \max_{i,j} r_{ij}$ (so for APPROX-k-VERTEX-CONN, $r_{\max} = k$ and for APPROX-0,1,2-SNDP, $r_{\max} = 2$) and let $m' = \min(m, r_{\max} n)$. It will take $O(r_{\max} m')$ total time per vertex pair to solve the initial flow problem at the beginning of AUGMENT: there are at most m' edges in I and we need to find at most $r_{\max}$ augmenting paths. If there are more than $r_{\max}$ augmenting paths, then the flow value is greater than $r_{\max}$ and there will be no violated set associated with the s–t flow. As we update the residual graph of the s–t flow over the course of the algorithm AUGMENT, if we find an additional augmenting path, then there will be no further violated sets associated with the s–t flow. Hence the total time taken to update the residual graph for an s–t pair is $O(m')$ time. Thus finding active sets will take $O(r_{\max} m' n^2 + n^3 + m' n^2) = O(r_{\max} m' n^2)$ time per call to AUGMENT.

To implement the edge selection step, we keep track of a variable $d(e) = \sum_{S: e \in \delta(S: \zeta_{I \cup F}(S))} y_S$ for each edge e. Let $a(e)$ denote the number of sets $C \in \mathcal{C}$ for which $e \in \delta(C : \zeta_{I \cup F}(C))$. Then in each iteration we search for the edge that minimizes $\varepsilon = (c_e - d(e))/a(e)$. Because of Theorems 3.5 and 3.7, we can prove that the active sets over all iterations of a phase form a laminar family: an active set in a future iteration cannot cross an active set in the present iteration since an active set in the future is violated in the present iteration. Thus we can use a union-find structure to keep track of the vertices in the current collection $\mathcal{C}$ of active sets. Whenever a new active set C is formed, we use $O(m' \alpha(n, n))$ time to find the vertices in $\Gamma_I(C)$. Then in each iteration we examine each edge to compute $(c_e - d(e))/a(e)$. This takes $O(\alpha(n, n) + r_{\max})$ time per edge: $O(\alpha(n, n))$ time to determine the $C \in \mathcal{C}$ to which its endpoints belong and $O(r_{\max})$ time to check if the edge is in $\delta(C : \zeta_{I \cup F}(C))$. It takes $O(n\alpha(n, n))$ time per call to AUGMENT to maintain the union-find structure on the active sets. Thus the overall running time of the edge selection process is $O(mn(\alpha(n, n) + r_{\max}))$ per call to AUGMENT.

Every time an edge is removed in the edge deletion stage, we must verify that the

remaining graph is still a feasible solution. In the case of the function h corresponding to phase p of APPROX-k-VERTEX-CONN, we must simply determine whether the graph is still p-vertex-connected. Steiglitz *et al.* [19] have shown that this can be done with $O(pn)$ network flows. Since each flow is in a graph with m' edges, and we need only p augmenting paths per flow, the time needed to compute each flow is $O(pm')$. We check $O(n)$ edges for deletion per call to AUGMENT, so that the total time used for the edge deletion step is $O(k^2m'n^2)$ per call to AUGMENT. In the case of the function h corresponding to the APPROX-0,1,2-SNDP algorithm, we can use the dynamic data structure of Rauch [17]. Given a graph, Rauch's data structure allows an edge insertion or deletion to occur in amortized $O(\sqrt{m}\log n)$ time, and can answer queries on pairs of vertices i, j in $O(1)$ time. The queries can be either "Are there two vertex-disjoint paths between i and j?" or "Are i and j connected?" Thus we can perform the edge deletion stage of APPROX-0,1,2-SNDP in $O(n(\sqrt{m}\log n + n^2)) = O(n^3)$ time.

Putting all of these bounds together, we can implement a call to AUGMENT in $O(k^2m'n^2)$ time for APPROX-k-VERTEX-CONN, and $O(n^3+mn\alpha(n,n))$ time for APPROX-0,1,2-SNDP. The call in APPROX-0,1,2-SNDP to the EDGE-SNDP algorithm can be implemented in $O(n^2\log n)$ time [4]. Thus the overall running time for APPROX-k-VERTEX-CONN is $O(k^3m'n^2)$ time, and for APPROX-0,1,2-SNDP is $O(n^3+mn\alpha(n,n))$ time. This yields the following theorem.

THEOREM 5.1. *For undirected graphs $G = (V, E)$ with nonnegative edge costs,* APPROX-k-VERTEX-CONN *is a $2\mathcal{H}(k)$-approximation algorithm for the minimum-cost k-vertex-connectivity problem running in $O(k^3m'n^2)$ time, and* APPROX-0,1,2-SNDP *is a 3-approximation algorithm for the {0,1,2}-survivable network design problem running in $O(n^3 + mn\alpha(n,n))$ time, where $n = |V|$, $m = |E|$, $m' = \min(kn, m)$, $\mathcal{H}(k) = 1 + \frac{1}{2} + \cdots + \frac{1}{k}$, and $\alpha(n,n)$ is the inverse Ackermann function.*

6. Concluding Remarks. It would be very interesting to extend these results to the general survivable network design problem. However, our results here depend quite heavily on either the uniformity of the problem (for the k-vertex-connectivity problem) or the structure inherent in low-connectivity graphs (for the {0, 1, 2}-survivable network design problem). We do not know how an uncrossing lemma along the lines of Lemma 3.6 can be proven in the general case. Of course, it is possible that some entirely new technique will succeed where these primal–dual techniques fail to work. It is a testimony to the power of these techniques, however, that they extend to vertex-connectivity problems.

Acknowledgments. We thank Michel Goemans for providing many comments on a previous version of this work. We also thank the referees for their useful comments.

References

[1] 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.

[2] K. P. Eswaran and R. E. Tarjan. Augmentation problems. *SIAM Journal on Computing*, 5:653–665, 1976.

[3] M. Goemans, A. Goldberg, S. Plotkin, D. Shmoys, E. Tardos, and D. Williamson. Improved approximation algorithms for network design problems. In *Proceedings of the 5th Annual ACM–SIAM Symposium on Discrete Algorithms*, pages 223–232, 1994.

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

[5] M. Grötschel, C. L. Monma, and M. Stoer. Computational results with a cutting plane algorithm for designing communication networks with low-connectivity constraints. *Operations Research*, 40:309–330, 1992.

[6] F. Harary. The maximum connectivity of a graph. *Proceedings of the National Academy of Sciences, USA*, 48:1142-1146, 1962.

[7] T. Hsu. On four-connecting a triconnected graph. In *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science*, pages 70–79, 1992.

[8] T. Hsu and V. Ramachandran. A linear time algorithm for triconnectivity augmentation. In *Proceedings of the 32nd Annual Symposium on Foundations of Computer Science*, pages 548–559, 1991.

[9] T. Jordán. On the optimal vertex-connectivity augmentation. *Journal of Combinatorial Theory, Series B*, 63:8–20, 1995.

[10] S. Khuller and B. Raghavachari. Improved approximation algorithms for uniform connectivity problems. *Journal of Algorithms*, 21:434–450, 1996.

[11] S. Khuller and R. Thurimella. Approximation algorithms for graph augmentation. *Journal of Algorithms*, 14:214–225, 1993.

[12] 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 Technical Report CS-92-30, Brown University.

[13] K. Menger. Zur allgemeinen Kurventheorie. *Fundamenta Mathematicae*, 10:96–115, 1927.

[14] M. Mihail, D. Shallcross, N. Dean, and M. Mostrel. A commercial application of survivable network design: ITP/INPLANS CCS Network Topology Analyzer. In *Proceedings of the 7th Annual ACM–SIAM Symposium on Discrete Algorithms*, pages 279–287, 1996.

[15] C. L. Monma and D. F. Shallcross. Methods for designing communication networks with certain two-connected survivability constraints. *Operations Research*, pages 531–541, 1989.

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

[17] M. Rauch. Improved data structures for fully dynamic biconnectivity. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*, pages 686–695, 1994. Submitted to *SIAM Journal on Computing*.

[18] R. Ravi and D. P. Williamson. An approximation algorithm for minimum-cost vertex-connectivity problems. In *Proceedings of the 6th Annual ACM–SIAM Symposium on Discrete Algorithms*, pages 332–341, 1995.

[19] K. Steiglitz, P. Weiner, and D. Kleitman. The design of minimal cost survivable networks. *IEEE Transactions on Circuit Theory*, 16:455–460, 1969.

[20] M. Stoer. *Design of Survivable Networks*. Lecture Notes in Mathematics, volume 1531. Springer-Verlag, Berlin, 1992.

[21] T. Watanabe and A. Nakamura. A minimum 3-connectivity augmentation of a graph. *Journal of Computer and System Sciences*, 46:91–128, 1993.

[22] D. P. Williamson. On the design of approximation algorithms for a class of graph problems. Ph.D. thesis, MIT, Cambridge, MA, September 1993. Also appears as Technical Report MIT/LCS/TR-584.

[23] D. P. Williamson and M. X. Goemans. Computational experience with an approximation algorithm on large-scale Euclidean matching instances. *INFORMS Journal on Computing*, 8:29–40, 1996.

[24] 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.