COMBINATORICA
Bolyai Society – Springer-Verlag

COMBINATORICA **21** (1) (2001) 39–60

A FACTOR 2 APPROXIMATION ALGORITHM FOR THE GENERALIZED STEINER NETWORK PROBLEM*

KAMAL JAIN

Received March 6, 1998

We present a factor 2 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, which is also known as the survivable network design problem. Our algorithm first solves the linear relaxation of this problem, and then iteratively rounds off the solution. The key idea in rounding off is that in a basic solution of the LP relaxation, at least one edge gets included at least to the extent of half. We include this edge into our integral solution and solve the residual problem.

1. Introduction

We consider the problem of finding a minimum-cost subgraph of a given graph such that the number of edges crossing each cut is at least a specified requirement. Formally, given an undirected multigraph $G = (V, E)$, a non-negative cost function $c : E \to \mathcal{Q}_+$, and a requirement function $f : 2^V \to \mathcal{Z}$, solve the following integer program (IP):

$$\min \sum_{e \in E} c_e x_e \tag{1}$$

subject to:

$$\forall S \subseteq V : \quad \sum_{e \in \delta_G(S)} x_e \geq f(S)$$

$$\forall e \in E : \quad x_e \in \{0, 1\}$$

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

Mathematics Subject Classification (2000): 68W25, 90C57

* Supported by NSF Grant CCR 9627308.

0209–9683/101/$6.00 ©2001 János Bolyai Mathematical Society

In this paper, we assume that we are given a separation oracle for the linear relaxation in which x_e's are allowed to take any fractional value between 0 and 1 (see LP 2).

We further assume that f is *weakly supermodular*, i.e., it satisfies:

1. $f(V)=0$
2. For every $A,B\subseteq V$, at least one of the following holds
   - $f(A)+f(B)\leq f(A-B)+f(B-A)$
   - $f(A)+f(B)\leq f(A\cap B)+f(A\cup B)$

The problem was first considered in [15] with the stronger assumption that f is proper (see Definition 2.1). The authors of [15] give a $2k$-approximation algorithm, where k is the maximum requirement of a set. The approximation factor was later improved to $2H_k$ in [5], where $H_k = 1+\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{k}$. The algorithm in [5] also works for weakly supermodular functions.

No better approximation factor was known even for *the generalized Steiner network problem*, which was the main motivation for studying this class of problems. In the generalized Steiner network 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 can be reformulated as IP 1 by taking $f(S)=\max_{i\in S,j\notin S} r_{ij}$. It was known that such an f is proper and hence weakly supermodular.

Moreover, a separation oracle for the generalized Steiner network problem can be constructed using the max-flow min-cut theorem. If the requirement of some pair i,j is not satisfied then the inequality corresponding to minimum i-j-cut is violated. Otherwise the solution is feasible. Hence the generalized Steiner network problem is a special case of the problem we are considering.

The factor was not any better for the version of the problem in which we are allowed to choose multiple copies of edges. This version is usually considered a specialization of the generalized Steiner network problem because we can make k copies of each edge, where k is the maximum requirement. Technically, k can be exponentially large, so we can not write those edges explicitly. Our algorithm does not need an explicit representation of edges and hence solves this version too.

The most general case with factor 2 was the *Steiner forest problem* ([1,6]), where $f(S)$ is 0-1 proper function. Hence our algorithm puts the generalized case at par with this special case.

Our algorithm falls into the class of rounding algorithms. Rounding algorithms use an optimal fractional solution to obtain a good integral solution. Some problems, like vertex cover [10] and node multiway cut [4], have the remarkable property that they admit an optimal fractional solution which

is half-integral. When this property holds, rounding up gives an approximation factor of 2. Unfortunately, the half-integrality property does not hold for our problem: consider the Petersen graph with unit cost on edges and $r_{uv} = 1$ for every pair u and v. In any fractional solution, $x(\delta(v))$ is at least 1 for every vertex v. Hence, the cost of a fractional solution is at least 5. Since the Petersen graph is three edge connected, assigning $\frac{1}{3}$ to every edge will satisfy all the cut requirements. So, the cost of an optimal fractional solution is 5. If there is a half-integral optimal solution then the edges with $x_e = \frac{1}{2}$ will form a cycle. In fact, that cycle must cover all the vertices, for otherwise its vertex set will be an unsatisfied cut. It is well known that the Petersen graph does not have a Hamiltonian cycle.

Iterative rounding, which is introduced in this paper, does not need half-integrality. It is enough to find an optimal fractional solution, x, in which at least one edge e has x_e at least half. Such an x_e can be rounded to 1 while at most doubling its contribution to the cost of the solution. The part of cut requirements which is not satisfied by the integrally chosen edges defines a residual problem; this problem can also be modeled by IP 1. Assuming that we can find an optimal fractional solution for the residual problem in which at least one edge is picked to the extent of at least half, then, by iterating the rounding procedure, we get an approximation factor of 2. This is naturally extending the Rounding based algorithms, which as they were known, solve the LP once and obtain an integral solution by a suitable rounding process. Seemingly this method did not exploit the full power of linear programming. After part of the fractional solution had been rounded, the current solution might not be the best solution to continue with the rounding process. Iterative rounding round the fractional solution in phases. After each phase, it recomputes the best fractional solution, maintaining the rounding achieved in the previous phases.

The key result that makes iterative rounding work is that any basic feasible solution, x, of LP 2 has an edge e with x_e at least half (Figure 2 demonstrates this fact in the Petersen graph). Using this result we get an approximation factor of 2, thus proving that the integrality gap of LP 2 is at most 2. It was already known that this integrality gap is not less than 2, so, the integrality gap of LP 2 is exactly 2. We leave open the developing of purely combinatorial constant factor algorithm for the problem.

2. Preliminaries

In this section, we briefly establish a few facts about weakly supermodular functions, submodular functions and proper functions. These facts can also be found in [5].

Definition 2.1. A function $f : 2^V \to \mathcal{Z}_+$ is *proper* if $f(V) = 0$ and the following two conditions hold

1. For every subset S of V, $f(S) = f(V - S)$.
2. For all disjoint subsets A and B of V, $f(A \cup B) \leq \max\{f(A), f(B)\}$.

Theorem 2.2. *([5]) Every proper function, f, is weakly supermodular.*

Proof. From the definition of proper functions, we have

1. $\max\{f(A-B), f(A \cap B)\} \geq f(A)$,
2. $\max\{f(B-A), f(A \cup B)\} = \max\{f(B-A), f(V-(A \cup B))\} \geq f(V-A) = f(A)$,
3. $\max\{f(B-A), f(A \cap B)\} \geq f(B)$, and
4. $\max\{f(A-B), f(A \cup B)\} = \max\{f(A-B), f(V-(A \cup B))\} \geq f(V-B) = f(B)$

By summing the two inequalities which involve the minimum of $f(A - B), f(B - A), f(A \cup B), f(A \cap B)$, we get the required result (for example, if $f(A-B)$ is minimum then we sum the first and the last inequalities). ∎

Definition 2.3. A function $f : 2^V \to \mathcal{Z}$ is *submodular* if $f(V) = 0$ and for every two sets $A, B \subseteq V$, the following two conditions hold

1. $f(A) + f(B) \geq f(A-B) + f(B-A)$.
2. $f(A) + f(B) \geq f(A \cap B) + f(A \cup B)$.

Lemma 2.4. *For any multigraph G, the function $|\delta_G(.)|$ is submodular.*

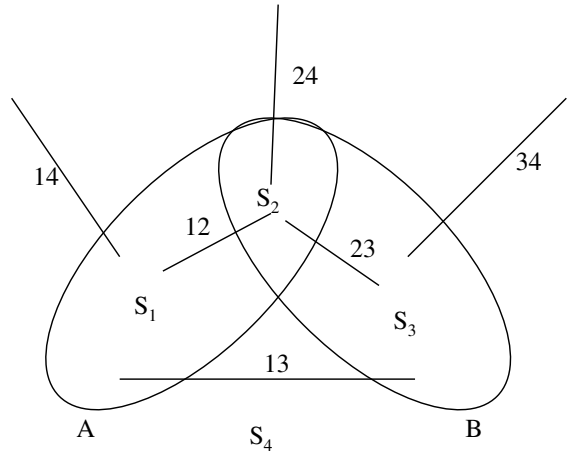

Fig. 1. Line labeled ij represents all the edges between sets S_i and S_j.

Proof. Let A and B be arbitrary subsets of V. For simplicity, denote the four quadrants by $S_1 \equiv A-B$, $S_2 \equiv A \cap B$, $S_3 \equiv B-A$ and $S_4 \equiv V-(A \cup B)$. Let $x(S_i, S_j)$ be the number of edges of x which have one end point in S_i and the other in S_j. The lemma is clear by observing the following (see Figure 1):

1. $|\delta_G(A)| = x(S_1,S_3)+x(S_1,S_4)+x(S_2,S_3)+x(S_2,S_4)$.
2. $|\delta_G(B)| = x(S_1,S_2)+x(S_1,S_3)+x(S_2,S_4)+x(S_3,S_4)$.
3. $|\delta_G(A-B)| = x(S_1,S_2)+x(S_1,S_3)+x(S_1,S_4)$.
4. $|\delta_G(B-A)| = x(S_1,S_3)+x(S_2,S_3)+x(S_3,S_4)$.
5. $|\delta_G(A \cap B)| = x(S_1,S_2)+x(S_2,S_3)+x(S_2,S_4)$.
6. $|\delta_G(A \cup B)| = x(S_1,S_4)+x(S_2,S_4)+x(S_3,S_4)$.

Theorem 2.5. ([5]) *Let G be a graph. Let $x \in \{0,1\}^{E(G)}$ be any subgraph of G. If $f : 2^{V(G)} \to \mathcal{Z}$ is a weakly supermodular function, then $f(S) - |\delta_x(S)|$ is also a weakly supermodular function.*

Proof. The theorem follows from the fact that $|\delta_x(.)|$ is submodular. ∎

3. A factor of 2 approximation algorithm

In this section we will show how to take a fractional solution to the following LP and round it to an integral solution whose objective value is within a factor of two of the objective value of the LP. This LP is in fact the linear relaxation of IP (1).

$$\min \sum_{e \in E(G)} c_e x_e \tag{2}$$

subject to:

$$\forall S \subset V : \quad \sum_{e \in \delta_G(S)} x_e \geq f(S)$$
$$\forall e \in E(G) : \quad 1 \geq x_e \geq 0$$

where G is a multigraph and f is a weakly supermodular function. We also assume that f takes a positive value somewhere.

To get a factor of 2 integral solution, we first solve the LP (2) fractionally; then we round off the solution iteratively using the following theorem.

Theorem 3.1. *In any basic solution to LP (2), for at least one edge, e, x_e is at least $\frac{1}{2}$.*

We will prove this theorem in the next section. Now let us see how this theorem helps us to iteratively round off the solution.

Let x^* be some optimal basic solution of the LP (2). Let $E_{\frac{1}{2}+}$ be the set of edges which took value at least $\frac{1}{2}$ in the solution x^*. Suppose $G_{res} = G - E_{\frac{1}{2}+}$. Now consider the LP, which is the residual LP, obtained from LP (2) by fixing the values of all the edges in $E_{\frac{1}{2}+}$ to 1.

$$\min \sum_{e \in E(G_{res})} c_e x_e \tag{3}$$

subject to:

$$\forall S \subset V : \quad \sum_{e \in \delta_{G_{res}}(S)} x_e \geq f(S) - \sum_{e \in E_{\frac{1}{2}+} \cap \delta_G(S)} 1$$

$$\forall e \in E(G_{res}) : \quad 1 \geq x_e \geq 0$$

From Theorem 2.5, $f(s) - \sum_{e \in E_{\frac{1}{2}+} \cap \delta_G(S)} 1$ is weakly supermodular; hence, LP (3) has the same form as LP (2).

Theorem 3.2. *Let optimal value of LP (2) and LP (3) be z^* and z^*_{res} respectively. If E_{res} is an integral solution to LP (3) with value at most $2z^*_{res}$, then $E_{res} \cup E_{\frac{1}{2}+}$ is an integral solution to LP (2) with value at most $2z^*$.*

Proof. Clearly, $E_{res} \cup E_{\frac{1}{2}+}$ is a feasible integral solution to LP (2). Notice that the restriction of x^* to G_{res} is a feasible solution for LP (3), hence we have

$$z^*_{res} \leq z^* - \sum_{e \in E_{\frac{1}{2}+}} x^*_e c_e, \tag{4}$$

i.e.,

$$2z^* \geq 2z^*_{res} + \sum_{e \in E_{\frac{1}{2}+}} 2x^*_e c_e.$$

We know that for every $e \in E_{\frac{1}{2}+}$, $2x^*_e \geq 1$, hence we have

$$2z^* \geq 2z^*_{res} + \sum_{e \in E_{\frac{1}{2}+}} c_e.$$

We also know that $\sum_{e \in E_{res}} c_e \leq 2z^*_{res}$, hence we have

$$2z^* \geq \sum_{e \in E_{res}} c_e + \sum_{e \in E_{\frac{1}{2}+}} c_e. \qquad \blacksquare$$

Now we have a high level description of a 2-approximation algorithm:

Algorithm *Iterative Rounding*

1. Find an optimal basic solution to LP (2).
2. Include all those edges which have values $\frac{1}{2}$ or more, in the solution.
3. Delete all the edges, which have been included in the solution, from the graph; and solve the residual problem.

Note that the LP's involved have exponentially many constraints. So we need to produce a separation oracle for LP (3), so that we can solve it in polynomial time using the ellipsoid algorithm ([9]). We also need to show that, once we have an optimal solution, we can find an optimal basic solution.

Suppose we have a separation oracle for LP (2). Let x_{res} be some vector in $[0,1]^{E(G_{res})}$ for which we either want to determine whether x_{res} is a feasible point for LP (3) or, if not, then we want to produce a constraint of LP (3) which is not satisfied.

Let us extend the vector x_{res} from $[0,1]^{E(G_{res})}$ to $[0,1]^{E(G)}$, by assigning 1 to the fields corresponding to the edges in $E_{\frac{1}{2}+}$. Let us call the extended vector x.

Clearly, for any set S,

$$x_{res}(\delta_{G_{res}}(S)) \geq f(S) - \sum_{e \in E_{\frac{1}{2}+} \cap \delta_G(S)} 1$$

if and only if

$$\sum_{e \in \delta_G(S)} x_e \geq f(S).$$

And hence, if x_{res} is feasible for LP (3) then x is feasible for LP (2); and if x_{res} violates any constraint in LP (3) then x violates the same constraint in LP (2). So given a separation oracle for LP (2), we can obtain a separation oracle for LP (3). From the assumption made, a separation oracle for LP (2) is given. Hence LP (3) belongs to the class of problem we are solving here.

To complete the algorithm, we need to show that, how an optimal solution can be converted into an optimal basic solution. Now let us say we have some optimal solution, x^*, which is not a basic solution to LP (2). In that case we can get one more inequality tight. Hence at some point we will get an optimal basic solution.

Let $\mathcal{E}$ be the set of inequalities which are tight, i.e., are satisfied by x^* as equality. Since x^* is not a basic solution, the affine subspace which the equalities in $\mathcal{E}$ define is at least one dimensional. Hence, it contains a line, L, passing through x^*. Because of constraints, $1 \geq x_e \geq 0$, polytope defined by LP (2) is bounded. Hence L has only a finite segment common with the polytope. Either end of that segment satisfies all the equalities in $\mathcal{E}$ plus at least one more, which is linearly independent of the equalities in $\mathcal{E}$. Algorithmically,

1. $\mathcal{E} \leftarrow \emptyset$
2. While x^* is not a basic solution do
   (a) Find a line passing through x^* in the affine subspace defined by $\mathcal{E}$.
   (b) Find one of the end point of the segment defined by the intersection of the line and the polytope of LP (2). Replace x^* by that point.
   (c) That end point makes one more inequality tight. Since this new equality was not tight earlier, this is linearly independent of equalities in $\mathcal{E}$. Include this equality in $\mathcal{E}$. This will increase the dimension of $\mathcal{E}$ which is bounded above by the number of edges.

Lemma 3.3. *Any optimal solution can be converted into an optimal basic solution in polynomial time.*

Theorem 3.4. *Given a separation oracle for the LP (2), algorithm* Iterative Rounding *described in this section, solves the LP integrally within twice the fractional optimal.*

4. Proof of Theorem 3.1

In this section we will prove Theorem 3.1.

Let x be some basic feasible solution. If there is some edge which takes the value 1, then the theorem holds trivially. So assume that no edge takes the value 1. Also, if some edge takes the value zero, then we can assume that the edge was never there. By assuming that we do not increase the cost of the optimal fractional solution. Hence, we may assume that no edge takes the value 0 either, i.e., x is totally fractional.

For notational convenience we represent the row of the constraint matrix corresponding to a set S by $\mathcal{A}_G(S)$. Let $x(A,B)$ represent the sum of all x_e's, where e has one end in A and the other in B.

We say that a set A is *tight* if $\sum_{e \in \delta_G(A)} x_e = f(A)$.

Lemma 4.1. *If two sets A and B are tight then at least one of the following must hold*

1. *$A-B$ and $B-A$ are also tight and $\mathcal{A}_G(A)+\mathcal{A}_G(B)=\mathcal{A}_G(A-B)+\mathcal{A}_G(B-A)$.*
2. *$A\cap B$ and $A\cup B$ are also tight and $\mathcal{A}_G(A)+\mathcal{A}_G(B)=\mathcal{A}_G(A\cap B)+\mathcal{A}_G(A\cup B)$.*

Proof. Denote the four quadrants by $S_1=A-B$, $S_2=A\cap B$, $S_3=B-A$ and $S_4=V-(A\cup B)$. Since A and B are tight we have (see Figure 1)

$$f(A) = x(S_1, S_3) + x(S_1, S_4) + x(S_2, S_3) + x(S_2, S_4),$$

and

$$f(B) = x(S_1, S_2) + x(S_1, S_3) + x(S_2, S_4) + x(S_3, S_4).$$

Since the solution is feasible, we also have

$$f(S_1) \leq x(S_1, S_2) + x(S_1, S_3) + x(S_1, S_4),$$

and

$$f(S_3) \leq x(S_1, S_3) + x(S_2, S_3) + x(S_3, S_4).$$

Since f is weakly supermodular, either $f(A)+f(B)\leq f(A-B)+f(B-A)$ or $f(A)+f(B)\leq f(A\cap B)+f(A\cup B)$. If the former holds then $x(S_2,S_4)=0$ and S_1 and S_3 are both tight. By our assumption, every edge has a non-zero value, so, $x(S_2,S_4)$ is 0 only if there is no edge between S_2 and S_4. In that case $\mathcal{A}_G(A)+\mathcal{A}_G(B)=\mathcal{A}_G(A-B)+\mathcal{A}_G(B-A)$.

Similarly, we can prove the second case in the lemma when $f(A)+f(B)\leq f(A\cap B)+f(\cup B)$. ∎

Let us denote the family of all tight sets by $\mathcal{T}$. For any family, $\mathcal{F}$, of tight sets, we denote the vector space spanned by the vectors $\mathcal{A}_G(S)$, $S\in\mathcal{F}$, by $Span(\mathcal{F})$. We say that two sets A and B *cross* if none of the sets $A-B$, $B-A$, and $A\cap B$ is empty. We say that a family of sets is *laminar* if no two sets in it cross.

Lemma 4.2. *For any maximal laminar family, $\mathcal{L}$, of tight sets, $Span(\mathcal{L})=Span(\mathcal{T})$.*

Proof. Since $\mathcal{L}\subseteq\mathcal{T}$, $Span(\mathcal{L})\subseteq Span(\mathcal{T})$. If the converse is not true then there exists a set S in $\mathcal{T}$ such that $\mathcal{A}_G(S)\notin Span(\mathcal{L})$. Choose one such S which crosses the minimum number of sets in $\mathcal{L}$.

Since, $\mathcal{A}_G(S)\notin Span(\mathcal{L})$, $S\notin\mathcal{L}$. From the maximality of $\mathcal{L}$, S must cross some set in $\mathcal{L}$. Let L be one of those sets. From the Lemma 4.1 one of the following must hold

1. $S-L$ and $L-S$ are also tight and $\mathcal{A}_G(S)=\mathcal{A}_G(S-L)+\mathcal{A}_G(L-S)-\mathcal{A}_G(L)$
2. $S\cap L$ and $S\cup L$ are also tight and $\mathcal{A}_G(S)=\mathcal{A}_G(S\cap L)+\mathcal{A}_G(S\cup L)-\mathcal{A}_G(L)$

Let us consider the first case; the second is similar. Since $\mathcal{A}_G(S) \notin Span(\mathcal{L})$, either $\mathcal{A}_G(S-L) \notin Span(\mathcal{L})$ or $\mathcal{A}_G(L-S) \notin Span(\mathcal{L})$. Again, since the two cases are similar, we consider the former only.

We claim that any set from the laminar family which crosses $S-L$ also crosses S. The fact that L crosses S but does not cross $S-L$ contradicts the choice of S.

Suppose $L' \in \mathcal{L}$ and L' crosses $S-L$. From set theory we get the following two implications.

1. $(S-L) \cap L' \neq \emptyset \Rightarrow S \cap L' \neq \emptyset$ and $L' - L \neq \emptyset$
2. $(S-L) - L' \neq \emptyset \Rightarrow S - L' \neq \emptyset$

Since left hand sides of both implications are true, we get $S \cap L' \neq \emptyset$, $L' - L \neq \emptyset$, and $S - L' \neq \emptyset$.

Since L and L' both belong to a laminar family, they do not cross. So either $L \subseteq L'$ or $L \cap L' = \emptyset$. We claim that in both cases $L' - S \neq \emptyset$. This claim together with the facts $S \cap L' \neq \emptyset$ and $S - L' \neq \emptyset$ proves the crossing of L' and S.

If $L \subseteq L'$ then $L - S \subseteq L' - S$. Since L and S cross, $L - S \neq \emptyset$, so, $L' - S \neq \emptyset$. If $L \cap L' = \emptyset$ then $L' - S = L' - (S-L)$. Since L' and $S-L$ cross, $L' - (S-L) \neq \emptyset$, so, $L' - S \neq \emptyset$. ∎

Because x is a basic solution, the dimension of $Span(\mathcal{T})$ is $|E(G)|$. Since $Span(\mathcal{L}) = Span(\mathcal{T})$, it is possible to choose a basis for $Span(\mathcal{T})$ from the vectors in $\{\mathcal{A}_G(S) : S \in \mathcal{L}\}$. Let $\mathcal{B} \subseteq \mathcal{L}$ such that $\{\mathcal{A}_G(S) : S \in \mathcal{B}\}$ forms a basis for $Span(\mathcal{T})$. Hence we have the following lemma.

Lemma 4.3. *There exists a laminar family, $\mathcal{B}$, of tight sets satisfying the following:*

1. $|\mathcal{B}| = |E(G)|$.
2. *The vectors $\mathcal{A}_G(S)$, $S \in \mathcal{B}$ are independent.*
3. *For every set $S \in \mathcal{B}$, $f(S) \geq 1$.*

Proof. The first two parts follow from the previous discussion. For the third part, notice that if $f(S)$ is less than zero then S can not be tight; and if $f(S)$ is zero then $\mathcal{A}_G(S)$ is the zero vector, hence contradicting the second part of the lemma. ∎

We are now ready to prove the Theorem 3.1. But before that, to convey the idea in a simpler setting, we will first prove an approximation factor of 3:

Theorem 4.4. *There is a tight set with positive requirement and at most three edges incident on it. Hence at least one of those edges takes a value of at least $\frac{1}{3}$.*

Proof. Take a laminar family, $\mathcal{B}$, as given by Theorem 4.3. Form a directed forest, F, with node set $\mathcal{B}$ and an edge from W to U whenever U is the smallest set containing W. We say that U is the parent of W and W is a child of U. A parent-less node is called root and a child-less node is called leaf.

To avoid confusion, we are using the word "node" for the nodes of the forest, F, and are using the word "vertex" for the vertices of the graph, G. Nodes will be represented by capital letters because they are also the sets of vertices, whereas vertices will be represented by small letters only.

We say that an edge *crosses* a node if it has one end inside the node and the other outside it.

Every edge $e \equiv (u, v)$ has two *endpoints*, denoted by e_u and e_v. Since we have $|E(G)|$ edges, we have $2|E(G)|$ endpoints. We say that an endpoint is *incident* to a node, U, if U is the smallest set containing that endpoint. So an endpoint is incident to one node only.

Note that from Theorem 4.3, we have $|V(F)| = |E(G)|$, i.e., the number of endpoints is exactly twice the number of nodes in the forest. Assume that Theorem 4.4 is not true. Under this assumption we will distribute the endpoints to the nodes of the forest such that every node gets at least two endpoints and every root at least four endpoints. This contradicts the equality $|V(F)| = |E(G)|$.

The subtree of F rooted at the node R consists of R and all its descendants. We will do the distribution inductively on every rooted subtree of F. An endpoint which is incident to any node in the subtree is said to be contained in the subtree.

Lemma 4.5. *For any rooted subtree of the forest, we can distribute the endpoints contained in it such that every node gets at least* 2 *endpoints and the root gets at least* 4.

Proof. If a leaf node has three or less edges crossing it, then one of those edges will take a value of at least $\frac{1}{3}$. So we can assume that each leaf has at least four endpoints in it. So the lemma is true if the subtree is just a leaf node of F.

Consider a subtree rooted at R. If R has two or more children then by induction each child gets at least four endpoints, and each of their descendents gets at least two endpoints. We will re-distribute the endpoints. R takes two endpoints from each of its children; they are still left with at least two endpoints each. Moreover, since R has two or more children, it now has at least four endpoints.

The only case that remains is when R has exactly one child. Let C be the only child of R. By induction, C gets at least four endpoints and each of its

descendents at least two. Since C has a surplus of at least two endpoints, we can re-distribute and assign that surplus to R. If R had two more endpoints of its own, i.e., endpoints which were incident to R, then the lemma follows. So, we may assume that R has at most one endpoint incident to it.

Since $\mathcal{A}_G(R)$ and $\mathcal{A}_G(C)$ are different vectors, there should be at least one edge which crosses C but not R, or else crosses R but not C. In both cases there will be an endpoint incident to R. Since we have assumed that R has at most one end point incident to it, R has exactly one endpoint incident to it.

The value of the edge which is giving one endpoint incident to R is the difference between the requirements of R and C. But this has to be an integer and we have assumed that the value of every edge is strictly fractional. This gives a contradiction. ∎

To prove the Theorem 3.1, let us assume the contrary, i.e., every edge takes a value strictly less than half. For this proof instead of working with the value of edges we will be working with their half complements, i.e., $y_e = \frac{1}{2} - x_e$. Clearly y_e's lie strictly between zero and half.

For any tight set, S, let us define its *co-requirement* as the sum of y_e's for all the edges crossing that set. It is easy to see that the co-requirement is $\sum_{e \in \delta_G(S)} y_e = \frac{1}{2}|\delta_G(S)| - f(S)$. Since $f(S)$ is integral, the co-requirement of S is integral or semi-integral (i.e., integral plus half) depending upon whether $|\delta_G(S)|$ is even or odd.

Similar to Lemma 4.5 we can prove the following lemma, which will prove Theorem 3.1.

Lemma 4.6. *For any rooted subtree of the forest, we can distribute the endpoints contained in it such that every vertex gets at least* 2 *endpoints and the root gets at least* 3*. Moreover, the root gets exactly* 3 *endpoints only if its co-requirement is half.*

The proof of this lemma is by case analysis. Many cases in the proof are abstracted as the next lemma by Vazirani [14].

Lemma 4.7. ([14]) *Suppose a node S has α children and has β endpoints incident to it, where $\alpha + \beta = 3$. Furthermore, each child of S, if any, has a co-requirement of half. Then the co-requirement of S is half.*

Proof. Since each child of S has co-requirement of half, each child has an odd degree. Using this and the fact that $\alpha + \beta = 3$, number of crossings of edges by S or its children is odd. Since an edge which does not have an endpoint incident to S crosses either none or two children, degree of S is odd. Hence

its co-requirement is semi-integral. Next, we show that co-requirement of S is strictly smaller than three halves, thereby proving the lemma. Clearly,

$$coreq(S) = \sum_{e \in \delta_G(S)} y_e \leq \sum_{S'} coreq(S') + \sum_{e} y_e,$$

where the first sum is over all children S' of S, and the second sum is over all edges e having an endpoint in S. Since both the sums together have exactly three terms and the terms in the first sum are halves and in the second sum are strictly less than half, if second sum is over a non-empty set then $coreq(S) < \frac{3}{2}$. So we may assume that $\beta = 0$ i.e. there are no endpoints incident to S. In this case all edges incident to the children of S cannot also be incident to S, since otherwise the vector corresponding to S will be the sum of the vectors corresponding to its children. Therefore,

$$coreq(S) < \sum_{S'} coreq(S') = \frac{3}{2}$$ ∎

Proof of Lemma 4.6 A leaf node must have at least three edges crossing it otherwise one of the edge takes a value of at least half. In case exactly three edges cross the leaf then by Lemma 4.7 co-requirement of it is half. So, the lemma is true for all the subtree consists of only a leaf node.

Now consider a subtree rooted at R. If R has four or more children then, by induction, each child has at least three endpoints. So, R can take one endpoint from the surplus of each child, thus getting at least four endpoints. It remains to consider the cases when R has three or fewer children.

R has three children: If one of them has a surplus of two then R can get four endpoints from its children. R can also get four endpoints in case there is an endpoint incident to it. So we may assume that all the three children have surplus of exactly one and there is no endpoint incident to R. So R can take three endpoints from the surplus of its children and by Lemma 4.7 its co-requirement is half.

R has two children, C_1 and C_2: If each child has a surplus of at least two then R can take two endpoints from each of them. So we may assume that at least one child, say C_1, has a surplus of exactly one. Hence the co-requirement of C_1 is half. Now, we claim that R has an endpoint incident to it. By means of contradiction, let us assume the contrary.

Let α denote the number of edges running between C_1 and C_2. Since there is no endpoint incident to R,

$$|\delta_G(R)| = |\delta_G(C_1)| + |\delta_G(C_2)| - 2\alpha.$$

Since the co-requirement of C_1 is half, $|\delta_G(C_1)|$ is odd. So, the parity of $|\delta_G(R)|$ is different from the parity of $|\delta_G(C_2)|$.

Again, since there is no endpoint incident to R, the co-requirements of R and C_2 can differ by at most the co-requirement of C_1, which is half. Because $\mathcal{A}_G(R) \neq \mathcal{A}_G(C_1) + \mathcal{A}_G(C_2)$, there should be an edge between C_1 and C_2. This implies that the co-requirement of R cannot be half more than the co-requirement of C_2. Also, because $\mathcal{A}_G(C_2) \neq \mathcal{A}_G(R) + \mathcal{A}_G(C_1)$, there should be an edge which is crossing both C_1 and R. So, the co-requirement of R cannot be half less than the co-requirement of C_2.

Because of the different parities of $|\delta_G(R)|$ and $|\delta_G(C_2)|$, the co-requirement of R and the co-requirement of C_2 cannot be same either. This contradiction shows that there is an endpoint incident to R. If one child has a surplus of two or there are two endpoints incident to R, then we can clearly give four endpoints to R. So, we may assume that each child has a surplus of exactly one endpoint and there is only one endpoint incident to R. In this case we assign three endpoints to R and by Lemma 4.7 its co-requirement is half.

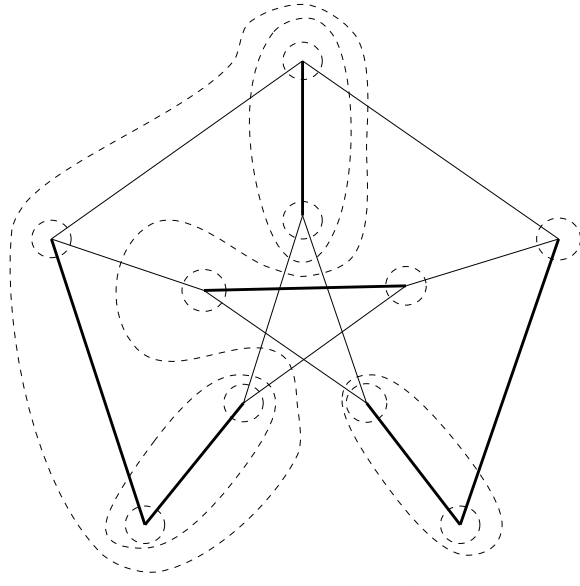

Fig. 2. A thick line represents an edge of value half. A thin line represents an edge of value quarter. The edge which is missing has a value of zero. Dashed lines represent a laminar family of tight sets which defines a basis for the solution.

R has only one child, C: If there are at least three endpoints incident to R, then we can assign one more endpoint to it from the surplus of C. We already have proved in Lemma 4.5 that R has at least two endpoints incident to it. So, we may assume that R has exactly two endpoints incident to it. If C has a surplus of at least two then also R can get four endpoints. So, we may assume that C has a surplus of exactly one and so its co-requirement is half. In this case we can assign three endpoints to R and by Lemma 4.7 its co-requirement is half. ∎

5. An example

Consider the Petersen graph with unit cost on edges and $r_{uv} = 1$ for every pair u,v. An optimal basic solution is shown in Figure 2. There are fifteen such possible solutions. The uniform solution, where every edge has a value one third, is the average of those fifteen optimal basic solutions.

6. Tight example

Consider the wheel on the vertex set $\{c, a_1, a_2, \ldots, a_n\}$, where c is the center of the wheel. All the edges incident on the center of the wheel are of length 1, whereas all others are of cost $2-\epsilon$. The requirement of each pair, a_i, a_j is 1 and between c and any other node is 0. The solution given by the algorithm is of the cost $(n-1)(2-\epsilon)$, while the optimum is of cost n only.

7. Running time improvement

The algorithm stated in Section 3 solves an LP at each iteration. In this section we will show that this is not necessary. The solution from one iteration can be used to compute a basic feasible solution for the next iteration. Details of this are given in the next section, here we describe the high level idea.

Note that to prove Theorem 3.2 we just need inequality (4), and to prove Theorem 3.1 we just need a basic feasible solution (not necessary optimal) of LP (3).

Let us denote the projection of x^* on G_{res} by $x_{G_{res}}$. We know that $x_{G_{res}}$ is a feasible solution for the LP (3) and also satisfies inequality (4). We will next show how to convert $x_{G_{res}}$ into a basic solution of lower or the same cost, using an algorithm similar to the algorithm which we used in the proof of Lemma 3.3.

If $x_{G_{res}}$ is not a basic solution, then the affine subspace defined by the equalities satisfied by $x_{G_{res}}$ has dimension at least one. Hence this affine subspace contains a line passing through $x_{G_{res}}$. Since the feasible region is bounded, the part of the line contained in the feasible region is also bounded. Each of the end points of that line segment tightens at least one more inequality. Since the objective function is linear, at least one of the two end points does not increase the cost of the solution. We replace $x_{G_{res}}$ by this end point. Hence we get one more tight inequality and still keep the inequality (4) satisfied. We can repeat this procedure until we get a basic feasible solution.

8. Implementing the algorithm

We are restating the problem so that it includes both the version in which multiple copies of an edge are allowed and the version in which they are not. Given a connected undirected multigraph $G = (V, E)$, a non-negative cost function $c : E \to \mathcal{Q}_+$, an availability function $a : E \to \mathcal{Z}_+ \cup \{\infty\}$, and a weakly supermodular requirement function $f : 2^V \to \mathcal{Z}$, solve the following IP

$$\min \sum_{e \in E} c_e x_e \tag{5}$$

subject to:

$$\forall S \subseteq V : \quad \sum_{e \in \delta_G(S)} x_e \geq f(S)$$
$$\forall e \in E : \quad x_e \in \{0, 1, \ldots, a_e\}.$$

Let n be the number of vertices and m be the number of edges in the graph. Let T be the maximum size of the numbers involved, if they are represented in binary.

We first solve the linear relaxation of this IP by Vaidya's algorithm ([13]), for which we need a separation oracle. We will give a separation oracle for the linear relaxation of the problem, under the assumption that $f(S) = \max_{i \in S, j \notin S} r_{ij}$, for some $r : V \times V \to \mathcal{Z}_+$.

Let $x \in \mathcal{R}_+^E$ be a given vector such that, for every e, $x_e \leq a_e$. We want to find whether x is feasible for the linear relaxation of IP (5) or not; if not then we want to find a set, S, such that the constraint corresponding to S is not satisfied. Note that x is infeasible if and only if there is a pair (i, j) of vertices such that the capacity of minimum i-j-cut is strictly less than r_{ij}, where the capacity of an edge e is x_e. Moreover, if there exist such a pair (i, j) then any minimum i-j-cut is an unsatisfied set.

So we have to check whether the capacity of minimum i-j-cut is at least r_{ij}, for all pairs (i,j). The best way to do this is through Gomory-Hu cut ([7]) trees. Computing a Gomory-Hu cut tree can be done with $n-1$ max-flow computations. So, a Gomory-Hu cut tree can be found in $O(n)M(m,n)$, where $M(m,n)$ is the time taken by one max-flow computation. After computing a Gomory-Hu cut tree, we can find a minimum i-j-cut in $O(n)$. Since there are $O(n^2)$ pairs we can do the checking in $O(n^3)$ time.

Hence we obtain a separation oracle that runs in $O(n)M(m,n)+O(n^3)=O(n)M(m,n)$ time. By plugging this in the running time of Vaidya's algorithm [13], it follows that an optimal solution to the linear relaxation of IP (5) can be found in $O(m^2n(T+\log m))M(m,n)+O(m^2(T+\log m))P(m)$ time, where $P(m)$ is the time to multiply two $m\times m$ matrices.

In the previous section we gave a high-level description of the algorithm that converts a feasible solution x into a basic feasible solution of lower or same cost. The detailed description follows:

1. $\mathcal{E}\leftarrow\emptyset$; $\mathcal{S}\leftarrow\mathcal{R}^m$
2. While $\dim(S)>0$ do
   (a) We will maintain the invariant that any point in $\mathcal{S}$ satisfies all the equalities in $\mathcal{E}$. To find a line passing through x in $\mathcal{S}$, let D be a vector orthogonal to every vector $\mathcal{A}_G(S)$, where $\mathcal{A}_G(S)=f(S)$ belongs to $\mathcal{E}$ (D can be found in $P(m)$ time). The line, $L(t)=x+tD$, passes through x and lies in $\mathcal{S}$.
   (b) Without loss of generality, assume that the cost of x decreases as we move on the line in the direction D (if this is not true then we can replace D by $-D$). Using binary search and the separation oracle, find the largest t for which $L(t)$ is a feasible solution. Denote the largest t by t^*. As explained below, we can find one inequality, $\sigma(x)\geq f$, not dependent upon the equalities in $\mathcal{E}$, such that $\sigma(L(t^*))=f$.
   (c) $x\leftarrow L(t^*)$
   $\mathcal{E}\leftarrow\mathcal{E}\cup\{\sigma\}$
   $\mathcal{S}\leftarrow\mathcal{S}\cap\{x\colon\ \sigma(x)=f\}$
   (Note that $\dim(\mathcal{S})$ decreases by one.)

A computation similar to that in the analysis of the ellipsoid algorithm [9] shows that we have to run the binary search in step (2.b) until we get two points $t_{low}<t_{up}$ such that $L(t_{low})$ is feasible, $L(t_{up})$ is infeasible, and $t_{up}-t_{low}<\frac{1}{m!2^T Poly(m)}$. Then, t^* will be the unique rational number with denominator bounded above by $m!2^T Poly(m)$. Moreover, we can take $\sigma(x)\geq t$ to be the unsatisfied inequality returned by separation oracle on $L(t^{up})$.

Since the initial interval for the binary search has length at most $\sqrt{m}$, the number of iterations (i.e., calls to the separation oracle) in the binary search is $O(m(T+\log m))$. So, each iteration of the loop in step (2) takes $P(m)+O(m(T+\log m))O(n)M(m,n)$ time. As the loop is repeated m times, the total time taken by the above algorithm is $O(m)P(m)+O(m^2n(T+\log m))M(m,n)$.

Now we will give the implementation of the main rounding algorithm. For any real number $I+f$, where I is an integer and f is a non-negative fraction, $[I+f]$ is defined as : I if f is strictly less than half and $I+1$ otherwise.

Solve the linear relaxation of IP (5) and then apply the above algorithm to obtain a basic feasible optimal solution, x^*.

1. $lp \leftarrow$ linear relaxation of IP (5).
2. For every e, $x_e^I \leftarrow 0$
3. $x^{lp} \leftarrow x$ (we will maintain the invariant that x^{lp} is a basic feasible solution to lp).
4. While x^I is not a feasible solution to IP (5), do
   (a) for every edge e, $x_e^I \leftarrow x_e^I + [x_e^{lp}]$, $x_e^{lp} \leftarrow \max\{0, x_e^{lp} - [x_e^{lp}]\}$
   (b) $lp \leftarrow$ residual LP.
   (c) Remove all edges e for which $x_e^{lp} = 0$.
   (d) Convert x^{lp} into a basic solution to lp, of same or lower value.

The loop in the algorithm is repeated $O(m)$ times and each iteration of the loop takes $O(m)P(m)+O(m^2n(T+\log m))M(m,n)$ time. Hence the total time taken by rounding is $O(m^2)P(m)+O(m^3n(T+\log m))M(m,n)$ time.

By taking into account the time taken by Vaidya's algorithm [13], the time taken by the implementation is $O(m^2(T+\log m))P(m)+O(m^3n(T+\log m))M(m,n)$.

9. Implementing the algorithm in strongly polynomial time

Tardos gave a strongly polynomial algorithm to solve combinatorial LPs in [12]. In that algorithm she requires an explicit declaration of all the constraints. If we can write LP (3) compactly with polynomial number of constraints then we can use Tardos' algorithm [12]. Note that LP (3) is a restriction of LP (2). It is obtained by taking a partial solution and then fixing the values of some of the edges in that partial solution. So, if we represent LP (2) compactly then we can do the same with LP (3).

In general we might not be able to represent LP (2) compactly. But we can do so in the special case when the function $f(S)$ is derived from requirements r_{ij} as in the last section. The LP relaxation of IP (5) is:

$$\min \sum_{e \in E} c_e x_e \tag{6}$$

subject to:

$$\forall S \subseteq V : \quad \sum_{e \in \delta_G(S)} x_e \geq \max_{i \in S, j \notin S} r_{ij}$$
$$\forall e \in E : \quad a_e \geq x_e \geq 0.$$

The idea to represent it compactly is as follows: we can consider x as the capacity function on the edges, i.e., x_e is the capacity of the edge e purchased at the cost of c_e per unit. We can purchase at most the capacity a_e on the edge e. Now we want to purchase capacities on edges so that we can transport at least r_{ij} unit of flow from i to j (we can pre-select the direction of flow required for each pair ij). This constraint can be written for each ij pair independently.

Suppose f_{ij} represents the flow of the commodity ij from i to j. Also suppose f_{ij}^{uv} represents the flow on the edge (uv) of the commodity ij from u to v, where (uv) represents the undirected edge between u and v.

The compact program is the following:

$$\min \sum_{e \in E} c_e x_e \tag{7}$$

subject to:

$$\forall e \in E : \quad a_e \geq x_e \geq 0$$

$\forall ij \in V \times V$ such that $r_{ij} > 0$ (note that the direction of flow required is pre-selected and hence $r_{ji} = 0$):

$$f_{ij} \geq r_{ij}$$
$$f_{ij} = \sum_{v \in V} f_{ij}^{iv}$$
$$\forall w \in V : \quad \sum_{u \in V} f_{ij}^{uw} = \sum_{v \in V} f_{ij}^{wv}$$
$$\forall (uv) \in E : \quad x_{(uv)} \geq f_{ij}^{uv} \geq 0.$$

By the maxflow-mincut theorem, the feasible region of LP (6) is same as the projection of the feasible region of LP (7) on the variables x_e.

By Tardos' result in [12], we can find an optimal solution to LP (7). But this solution may not be a basic solution to LP (6). The algorithm for converting it into a basic solution given in the previous section is not

strongly polynomial, so, we have to use a method that avoids the use of basic solutions.

First find any optimal solution, x^*, to LP (6) using the Tardos' algorithm on LP (7). Clearly, by modifying a_e to $\lceil x_e^* \rceil$ we do not increase the fractional optimum. So, after picking $\lfloor x_e^* \rfloor$ copies of each edge e, the residual problem is of the form IP (1) (i.e., each edge can be picked at most once). Theorem 3.1 shows that, for at least one edge e, we can add the constraint $x_e \geq \frac{1}{2}$ to the residual problem without increasing its optimal cost. We will simply try this for all edges.

So, we can find a factor 2 solution to IP (5) by solving LP (7) $O(m^2)$ times. Tardos' algorithm takes $O(n^{10}m^5)$ to solve LP (7) since it has $O(n^2m)$ variables. So, the total time for this implementation of the algorithm is $O(n^{10}m^7)$.

10. Discussion

Iterative rounding requires that in each iteration, one variable should take a value of at least half. This condition is strictly weaker than half integrality, in which all non-zero variables should take half integral values. Besides having larger applicability, iterative rounding might be also easier to establish than half integrality, like in the case of vertex cover problem.

Several aspects of iterative rounding remain to be explored. Right now, iterative rounding requires the solution of a linear program in each iteration. A more efficient way would be to choose the variables to be made 1 in the integral solution, through a combinatorial method, and then show the decrease in the optimal value of the residual LP.

This work shows that the integrality gap of LP 2 is 2. This fact makes the open problem of finding a purely combinatorial factor 2 approximation algorithm even more plausible. Integrality gap of LP 2 remains 2 even when modeling the Minimum Spanning Tree problem. Edmonds [3] gave an exact LP for this special case through the bidirected-cut formulation. It is believed that the LP obtained through this formulation for Steiner Tree problem has integrality gap close to 1. Recently, Rajagopalan and Vazirani [11] showed the integrality gap of this formulation to be 3/2 for quasi bipartite graphs, graphs which do not have edges running between two Steiner vertices.

Consider Node Connectivity problem, which is same as Generalized Network problem except that instead of edge disjoint paths we require node disjoint paths. The problem is essentially unsolved except for the case when the graph is unweighted and the requirement function is a constant [2]. The difficulty arises in defining the residual requirement of a set when a partial

solution is given. Recently, Jain et. al. defined Element Connectivity problem [8] which is same as Node Connectivity except that it allows two paths to intersect at required vertices (i.e., vertices with positive requirements). This relaxation allows them to define the residual requirement of a set, which enables them to give a primal-dual schema based approximation algorithm, using ideas from [15,5]. It will be interesting to see whether the ideas in this paper work for the Element Connectivity problem.

Acknowledgments I would like to thank my research advisor Vijay V. Vazirani, without whose help, this research would not have been possible. I would also like to thank Ion Măndoiu, who gave constructive comments during the research and on the writeup, to Michel Goemans, who gave the idea to represent the LP compactly, to Carl Burch, who gave the idea to present the Lemma 4.5 nicely, and to the unknown refree who gave helpful comments to improve the writeup.

References

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

[2] J. Cheriyan and R. Thurimella: Approximating minimum-size k-connected spanning subgraphs via matching, to appear in *SAIM J. Computing*.

[3] J. Edmonds: Optimum branchings, *J. Res. Nat. Bur. Standards*, **71** (1967), 233–240.

[4] N. Garg, V. V. Vazirani, and M. Yannakakis: Approximation algorithms for multiway cuts in node-weighted and directed graphs, *Proc. 21th International Colloquium on Automata, Languages and Programming*, 1994.

[5] M. X. Goemans, A. Goldberg, S. Plotkin, D. Shmoys, E. Tardos, and D. P. Williamson: Approximation algorithms for network design problems, *SODA*, 223–232, 1994.

[6] M. X. Goemans and D. P. Williamson: A general approximation technique for constrained forest problem, *SIAM J. Computing*, **24** (1995), 296–317.

[7] R. Gomory and T. Hu: Multi-terminal network flows, *SIAM J. Applied Mathematics*, **9** (1961), 551–570.

[8] K. Jain, I. Măndoiu, V. V. Vazirani, and D. Williamson: Primal dual schema based approximation algorithm for the element connectivity problem, *Proc. 10th annual Symposium on Discrete Algorithms*, 1999, 484–489.

[9] L. G. Khachiyan: A polynomial algorithm for linear programming (in Russian), *Doklady Akademiia Nauk* USSR **244** (1979), 1093–1096. A translation appears in: *Soviet Mathematics Doklady* **20** (1979), 191–194.

[10] G. L. Nemhauser and L. E. Trotter, Jr.: Vertex packing: structural properties and algorithms, *Mathematical Programming*, **8** (1975), 232–248.

[11] S. Rajagopalan and V. V. Vazirani: On the bidirected cut relaxation for the metric Steiner tree problem, *Proc. 10th annual Symposium on Discrete Algorithms*, 1999.

[12] É. Tardos: A strongly polynomial algorithm to solve combinatorial linear programs, *Operations Research*, **34** (1986), 250–256.
[13] P. M. Vaidya: A new algorithm for minimizing convex functions over convex sets, *Mathematical Programming*, **73** (1996), 291–341.
[14] V. V. Vazirani: *Approximation Algorithms*, Book in preparation. Available at http://www.cc.gatech.edu/fac/Vijay.Vazirani/book.ps, 2000.
[15] D. P. Williamson, M. X. Goemans, M. Mihail, and V. V. Vazirani: A primal-dual approximation algorithm for generalized Steiner network problems, *Combinatorica*, **15** (1995), 435–454.

Kamal Jain

College of Computing
Georgia Institute of Technology
kjain@cc.gatech.edu