An Integer Linear Programming Approach to the Steiner Problem in Graphs

Y. P. Aneja
School of Administration, University of New Brunswick, Canada

Consider a connected undirected graph $G[N; E]$ with $N = S \cup P$, the set of nodes, where P is designated as the set of Steiner points. A weight is associated with each edge e_i of the set E. The problem of obtaining a minimal weighted tree which spans the set S of nodes has been termed in literature as the Steiner problem in graphs. A specialized integer programming (set covering) formulation is presented for the problem. The number of constraints in this formulation grows exponentially with the size of the problem. A method called the row generation scheme is developed to solve the above problem. The method requires knowing the constraints only implicitly. Several other problems which can be put in a similar framework can also be handled by the above scheme. The generality of the scheme and its efficiency is discussed. Finally the computational result is demonstrated.

1. INTRODUCTION

Consider an undirected connected network $G[N;E]$ where N is the set of nodes and E denotes the set of m edges. Associated with each edge e_i of E is a weight $c_i > 0$. Let $T = T(N', E')$, where $N' \in N$ and $E' \in E$, be a tree associated with the graph. The weight of the tree is defined as

$$W(T) = \sum_{e_i \in E'} c_i$$

Let $N = S \cup P$, where P is designated as the set of Steiner points. A tree $T(N', E')$ where $S \subset N' \subset N$ and $E' \subset E$ is said to be an S-spanning tree associated with the network. The problem of determining an S-spanning tree $T^*(N^*, E^*)$ with minimum weight, is termed the Steiner problem in graphs. The problem is so named because of its obvious relation to the Steiner problem in two dimensional Euclidean space. Hakimi [5] has shown that if one can solve the Steiner problem on graphs, then one can solve various "cover problems" in graphs.

Clearly when $S = N$ the problem reduces to one of finding the minimal spanning tree and there exist efficient algorithms for doing so [1, 2]. For $|S| = 2$ one needs finding only the shortest chain between two nodes of S.

There are three known methods [3-5], for solving the Steiner network problem. All these employ specialized implicit enumeration techniques.

Below we give a set-covering formulation for the Steiner problem in graphs. Although the number of constraints in this formulation grows exponentially with the

0028-3045/80/0010-0167$01.20

size of the problem the method developed to solve the above set-covering problem is able to handle the constraints implicitly. This method, which we call the row generation scheme, is a modification of a known algorithm [7] for solving the general set-covering problem. Since the row generation scheme makes use of some properties of the coefficient matrix of the set-covering problems, we devote Sec. 3 to state these properties and the set-covering algorithm. Sections 4 and 5 develop the row generation scheme and shows how, by imbedding this scheme, the general set-covering algorithm of Bellmore and Ratliff could be modified to handle this special structured set-covering problem and various other problems which can be put in a similar framework. Section 6 explains how the relaxed linear program associated with the integer programming problem could be solved with only an implicit knowledge of the constraints. The linear programming solutions, though not necessary theoretically, are extremely useful from the computational point of view. Section 7 deals with some heuristic rules for improving the computational efficiency of the algorithm and Sec. 8 gives some computational results and conclusions derived by this study.

2. FORMULATION

2.1 Steiner Network Problem

Consider a partition of $N = X \cup \bar{X}$ such that $X \cap \bar{X} = \emptyset, S \cap X \neq \emptyset$ and $S \cap \bar{X} \neq \emptyset$. Let $P = (X, \bar{X})$ denote the cut-set of edges between X and $\bar{X}$. Clearly there are several such sets. Suppose we enumerate all such sets to be $P_1, P_2, \cdots, P_p$. Define constants

$$a_{ij} = \begin{cases} 1 & \text{if edge } e_j \in \text{cut set } P_i \\ 0 & \text{otherwise} \end{cases}$$

where $i = 1, 2, \cdots, p$ and $j = 1, 2, \cdots, m$. Consider now the following set-covering problem:

$$\text{Min} \sum_{j=1}^{m} c_j x_j$$

such that

$$\sum_{j=1}^{m} a_{ij} x_j \geqslant 1 \tag{1}$$

where $i = 1, 2, \cdots, p, x_j = 0$ or 1, and $j = 1, 2, \cdots, m$. Let $x^* = (x_1^*, x_2^*, \cdots, x_m^*)$ be the optimal solution to (1). Define a set T^* such that

$$T^* = \{e_j | x_j^* = 1\}.$$

It is not difficult to see that T^* yields the solution to the Steiner network problem. First we note that T^* does not contain any cycles since $c_j > 0$ for all edges of the network. Second, T^* contains only one connected component which connects all the nodes of S, for otherwise a constraint corresponding to some cut-set P_i would be violated. Finally, since T^* is obtained from the optimal solution to (1), T^* gives an S-spanning tree with minimum weight.

We formulate below some more problems which can be put in a framework similar to the Steiner network problem.

2.2 Construction of a Multicommodity Network†

Consider a directed network $G[N; E]$ where E is the set of feasible directed arcs. Let $c_j > 0$ be the cost of constructing the arc e_j. Let

$$S = \{s_1, s_2, \cdots, s_q\} \subset N$$

and

$$T = \{t_1, t_2, \cdots, t_q\} \subset N$$

be denoted as the source and the sink set, respectively. The problem is to choose a set of arcs from E, the set of feasible arcs, which when constructed would connect every (s_i, t_i) pair with at least one chain, and to do so with minimum total cost. To formulate this as a set-covering problem, let $C_1, C_2, \cdots, C_p$ be an enumeration of all the cuts which separate the corresponding source-sink pairs. Define

$$a_{ij} = \begin{cases} 1 & \text{if cut } C_i \text{ contains arc } e_j \\ 0 & \text{otherwise} \end{cases}$$

where $i = 1, 2, \cdots, p; j = 1, 2, \cdots, m$ and

$$x_j = \begin{cases} 1 & \text{if arc } e_j \text{ is constructed} \\ 0 & \text{otherwise.} \end{cases}$$

With these definitions, optimal solution to problem (1) yields the desired set of arcs.

2.3 Minimal Multicommodity Disconnecting Set [6]

Consider again a multicommodity directed network $G[N; E]$. Define c_j as the cost associated with destroying the arc $e_j \in E$. The set $D \subset E$, which when removed from the network would destroy all chains from sources to their respective sinks at a minimum cost is termed the minimal multicommodity disconnecting set. Defining

$$a_{ij} = \begin{cases} 1 & \text{if arc } e_j \in \text{chain } C_i \\ 0 & \text{otherwise} \end{cases}$$

†This problem was suggested by my colleague Professor K. P. K. Nair.

and

$$x_j = \begin{cases} 1 & \text{if } e_j \text{ is removed from the network} \\ 0 & \text{otherwise.} \end{cases}$$

The solution to (1) gives the desired set.

2.4 Maximal Acyclic Network

For a directed network $G[N; E]$ with c_j as defined in Sec. 2.3, one wants to obtain a set of arcs, at a minimum cost, which when removed from the network would make the resultant network acyclic. Suppose we enumerate $C_1, C_2, \cdots, C_p$ to be all the cycles in the network. Define

$$a_{ij} = \begin{cases} 1 & \text{if arc } e_i \in \text{cycle } C_j \\ 0 & \text{otherwise} \end{cases}$$

and

$$x_j = \begin{cases} 1 & \text{if arc } e_j \text{ is removed from the network} \\ 0 & \text{otherwise.} \end{cases}$$

With these interpretations of constants and variables, problem (1) yields the maximal acyclic graph.

Let us make some observations about these four problems. Clearly it is practically infeasible to enumerate all the constraints in (1) for any one of the above problems since the number of constraints grows exponentially with the size of the network. Thus, even for a moderate-size network, these problems can not be solved directly as set-covering problems using known set-covering algorithms [7, 8].

In this paper a method we call "Row Generation Scheme" is developed to solve problems of the above type without an explicit enumeration of the constraints in (1).

Since the row generation scheme makes use of some properties imbedded in the set-covering algorithm proposed by Bellmore and Ratliff [7], we devote the next section to these properties and their algorithm.

3. PROPERTIES OF THE SET-COVERING PROBLEM

Consider the linear program associated with the set-covering problem (1):

$$\text{Min} \sum_{j=1}^{m} c_j x_j$$

such that

$$\sum_{j=1}^{m} a_{ij} x_j \geqslant 1 \tag{2}$$

with $i = 1, 2, \cdots, p; x_j \geqslant 0; j = 1, 2, \cdots, m$. Let $x = (x_1, x_2, \cdots, x_m)$ be a solution to (1). With each solution is associated a set $J = \{j | x_j = 1\}$. This set J is called a *cover* of (1). A cover is called nonredundant if no proper subset of J is also a cover of (1).

Proofs of all the properties described below can be found in [9].

Property 1. Any optimal solution x^* to (2) satisfies $0 \leqslant x_j^* \leqslant 1, j = 1, 2, \cdots, m$; and $\langle x^* \rangle = (\langle x_1^* \rangle, \langle x_2^* \rangle, \cdots, \langle x_m^* \rangle)$, where $\langle x \rangle$ is the smallest integer greater than or equal to x, is a feasible solution to (1).

Property 2. Let $J = \{j_1, j_2, \cdots, j_k\}$ be a nonredundant cover of (1). Construct a $p \times k$ matrix $\bar{A}_1 = (\bar{a}_{pq})$ where $\bar{a}_{pq} = a_{pj_q}$. Then a suitable permutation of the rows of $\bar{A}$ yields a matrix $\bar{A}_1^P$ such that

$$\bar{A}_1^P = \left[\begin{array}{c} I \\ \hline R \end{array}\right]$$

where I is a $k \times k$ identity matrix.

Property 3. Given $J = \{j_1, j_2, \cdots, j_k\}$, a nonredundant cover to (1), we can construct a basis B for (2) and by a suitable permutation of B we can obtain a matrix

$$B^P = \left[\begin{array}{cc} I & 0 \\ \hline R & -I \end{array}\right].$$

Clearly $(B^P)^{-1} = B^P$, i.e., B^P is involutory. Also, if C_B is the price vector associated with columns of B then for any column a_j of the coefficient matrix, $z_j = C_B a_j^P$.

Property 4. Let B be the basis to problem (2) corresponding to a nonredundant cover to (1). A necessary condition that there exists a better feasible solution to (1) is that

$$\sum_{j \in Q} x_j \geqslant 1$$

where

$$Q = \{j | z_j - c_j > 0\}.$$

From the last property, we conclude that if $Q \neq \emptyset$, then we have an optimal solution to (1). Also it is important to note that the constraint in Property (4) is a set-covering constraint. Based on these properties, the following set covering algorithm was proposed by Bellmore and Ratliff.

Algorithm

Step (0): Let the original set covering problem be denoted as SC_1,

$$z = \sum_{j=1}^{m} c_j \text{ and } t = 1.$$

Step (1): Let $J^t = \{j_1, j_2, \cdots, j_k\}$ be any nonredundant cover to SC_t and B^P the corresponding involutory basis. If

$$\sum_{j=1}^{m} c_{j_i} < \bar{z},$$

then let

$$z = \sum_{i=1}^{k} c_{j_i}$$

and record J^t.

Step (2): Let $Q = \{j | c_B a_j^P - c_j > 0\}$. If $Q = \emptyset$ terminate; the last recorded cover yields an optimal cover. Otherwise append the cut-constraint

$$\sum_{j \in Q} x_j \geqslant 1$$

to SC_t and set $t = t + 1$. Go to Step (1).

The termination rule can be strengthened by the fact that if the optimal value of the linear program associated with SC_t is greater than or equal to the current best solution, then the current best solution is optimal to SC_1.

4. ROW GENERATION SCHEME

Suppose P is the permutation which yields matrices $\bar{A}_1^P$ and B^P, as defined in Property (2) and Property (3), respectively, corresponding to a nonredundant cover $J = \{j_1, j_2, \cdots, j_k\}$ of the set-covering problem (1). From Property (3), $z_j = c_B a_j^P$. But c_B can be written as $(c_{B_1} \vdots c_{B_2})$ where c_{B_1} is the price vector associated with variables $x_{j_1}, x_{j_2}, \cdots, x_{j_k}$; and c_{B_2} is the price vector associated with $\rho - k$ surplus variables. Now a_j^P, the jth column of the matrix A^P, can also be partitioned as

$$\begin{bmatrix} a_j^{(1)P} \\ \hdashline a_j^{(2)P} \end{bmatrix},$$

$a_j^{(1)P}$ being a column with k elements. Thus, noting that C_{B_2} is a vector of all zeros, we can write $z_j = C_{B_1} \cdot a_j^{(1)P}$. That is, in order to find the set $Q = \{j | z_j - c_j > 0\}$, we need to generate only the top k rows of the matrix $\bar{A}^P$.

This seemingly obvious result is extremely useful in the development of the row-generation scheme.

Consider, for instance, a problem which can be formulated as a set-covering problem (1). Assume that the coefficient matrix is not known explicitly but could be generated through some process which defines the constraints of the problem. Let us also assume that the constraints are too many in number to be dealt with explicitly. All four problems discussed in Sec. 2 fall in this category. Problem (1) can be solved without an explicit knowledge of these constraints if, at each iteration of the algorithm of Bellmore and Ratliff, we could obtain a different nonredundant cover to the problem and the set $Q = \{j | z_j - c_j > 0\}$. Given a cover, it should be easy to obtain a proper cover by going back to the original problem definition. It also follows from the result noted earlier, that given a nonredundant cover we can find the set Q if we can generate only the top k rows of the matrix A^P, k being the number of elements in the nonredundant cover. Let this nonredundant cover be $J = \{j_1, j_2, \cdots, j_k\}$. It follows from property (2) that the ith row α_i of the matrix A^P, $1 \leqslant i \leqslant k$, satisfies

$$\alpha_{ir} = \begin{cases} 1 & \text{if } r = j_i \\ 0 & \text{if } r \in J - \{j_i\}. \end{cases} \tag{3}$$

Thus any row of the coefficient matrix A which satisfies (3) can serve as the ith row of the matrix A^P. One would expect to generate one such row, with not much difficulty and without enumerating all the rows of A, since the process which generates all the constraints of the problem (1) is known.

In the next section we will illustrate clearly how the set Q could be generated, for the set-covering problem associated with the Steiner problem in graphs, at each step of the algorithm.

The computational efficiency or convergence of the set-covering algorithm discussed depends heavily on obtaining good lower and upper bounds to the optimal solution at each iteration. Computational experience with set-covering problems shows that linear programming is a very powerful tool for obtaining such bounds. Consider (2), the linear program associated with problem (1). The optimal solution to (2) clearly yields a lower bound for the optimal solution to (1). If, however, the optimal solution to (2) terminates all integer, then this is also the upper bound and we have the optimal solution to (1). If the optimal solution to (2) does not terminate integer, then one could use property 1 listed in Sec. 3 to obtain a cover and some heuristic rules to obtain a nonredundant cover from this cover which would serve as a good upper bound.

Obtaining a linear programming solution for (2) is no problem if the coefficient matrix is known explicitly. In the absence of explicit knowledge of the matrix A it is not clear how one would solve (2).

Consider the dual of (2):

$$\operatorname{Max} \sum_{i=1}^{\rho} U_i$$

such that

$$\sum_{i=1}^{\rho} a_{ij} U_i \leqslant c_j, \tag{4}$$

with $j = 1, 2, \cdots, m$; $U_i \geqslant 0$; $i = 1, 2, \cdots, \rho$.

Let us attempt to apply the simplex method to (4). The simplex method can be started by introducing all the slack columns into the basis. Let $\sigma_1, \sigma_2, \cdots, \sigma_m$ be the simplex multipliers associated with a basis at some iteration of the algorithm. We can assume that all $\sigma_i \geqslant 0$; for otherwise, if any multiplier is negative, the corresponding slack column can be introduced into the basis. In order to determine whether the basis is optimal, and if not, to determine the column of A^T which has to enter the basis, we have to find

$$\theta_0 = \min_k \sum_{i=1}^{m} \sigma_i a_{ki}. \tag{5}$$

If $\theta_0 \geqslant 1$, then the current basis is the optimal one, else we introduce the column, for which the minimum is attained, in to the basis to move to the next iteration of the simplex method. However, the columns of A^T, i.e., the rows of A are not known explicitly. Thus a row of A has to be generated for which the minimum in (5) is obtained.

We will show, in Sec. 6, how θ_0 can be obtained and a row generated for each of the problems in Sec. 2.

5. OBTAINING THE SET Q

We want to obtain the set $Q = \{j | z_j - c_j > 0\}$ for the set-covering problem (1) associated with the Steiner network problem, at each iteration of the set-covering algorithm. At iteration t, the problem SC_t has $p + t - 1$ constraints. The first p constraints are the original ones and the remaining $t - 1$ are the cuts added. Suppose $J = \{j_1, j_2, \cdots, j_k\}$ is a nonredundant cover to SC_t. Let us see first how to obtain the top row of the matrix A_t^P, A_t being the coefficient matrix of the problem SC_t.

First of all, check whether there exists a constraint p from the last $(t - 1)$ constraints such that $a_{pj_1} = 1$ and $a_{pj_r} = 0$ for $r = 2, 3, \cdots, k$, i.e., a row which satisfies (3). In other words, we check if any of the last $(t - 1)$ constraints would be violated by the solution:

$$x_j = \begin{cases} 1 & \text{if } j \in J - \{j_1\} \\ 0 & \text{otherwise.} \end{cases}$$

If any one of these constraints is violated, then the coefficient row corresponding to this constraint would serve as the top row of the matrix A_t^P. Otherwise, the above solution violates one of the first k constraints. Recall that these p constraints are needed to ensure that the solution would yield a subgraph which connects all the nodes of S. Thus the subgraph generated by the edges $e_{j_2}, e_{j_3}, \cdots, e_{j_k}$ has two connected components. Let $s_1 \in S$ and $s_2 \in S$ be the two nodes chosen from the first and second connected component, respectively. Consider now the network $G[N; E]$ with newly defined weights on edges as follows:

$$c_j^* = \begin{cases} c_j & \text{if } j \notin J \\ 0 & \text{if } j = j_1 \\ \infty & \text{if } j \in J - \{j_1\}. \end{cases} \tag{6}$$

Let $(X, \bar{X})$ be the min-cut separating s_1 and s_2 where c_j^* is defined as the capacity of the edge e_j. Clearly $e_{j_1} \in (X, \bar{X})$, for otherwise we have $e_j \notin (X, \bar{X})$ for $j \in J$, contradicting the fact that edges with indices in J yield a subgraph which connects s_1 and s_2. The coefficient row corresponding to this cut-set constraint clearly satisfies (3) for $i = 1$ and thus can be taken as the first row of the matrix A_t^P.

We can obtain similarly the remaining top $(k - 1)$ rows of A_t^P. Generating each row involves at most solving one max-flow problem. The set Q can thus be obtained.

It is not difficult to see how the set Q can be obtained if the set-covering formulation corresponded to any of the other three problems discussed in Sec. 2.2. For problem (2.2) it would involve finding min-cuts again, for each source-sink pair, over the network $G[N; E]$ with modified costs as defined in (6). We would need finding shortest cycles for maximal acyclic network problem, and shortest chains for each source-sink pair for the disconnective set problem discussed in Sec. 2.2. Efficient algorithms exist for performing all these tasks.

6. SOLVING THE LINEAR PROGRAM ASSOCIATED WITH SC$_t$

Similar to what we mentioned in Sec. 4, we could solve the linear program associated with SC$_t$ if for a given set of simplex multipliers $\sigma_1, \sigma_2, \cdots, \sigma_m$, all non-negative, we would find

$$\theta_0 = \operatorname*{Min}_k \sum_{i=1}^{m} a_{ki}\sigma_i.$$

Let

$$\theta_1 = \operatorname*{Min}_k \sum_{i=1}^{m} a_{ki}\sigma_i$$

with $k = 1, 2, \cdots, \rho$ and

$$\theta_2 = \operatorname*{Min}_k \sum_{i=1}^{m} a_{ki}\sigma_i$$

with $k = \rho + 1, \rho + 2, \cdots, \rho + t - 1$. Then $\theta_0 = \min(\theta_1, \theta_2)$. Now θ_2 can be determined easily since the last $t - 1$ constraints for SC$_t$ are known explicitly as they correspond to the cuts added. In order to obtain θ_1, let us interpret σ_i to be the weight associated with edge e_i. Then $\Sigma\, a_{ki}\sigma_i$ gives the total weight of the cut-set P_k. We want to obtain the cut-set with minimum weight. Let us assume that $(X, \bar{X})$ is one such cut-set which disconnects the set $S \cap X$ from the set $S \cap \bar{X}$. Let $s \in S \cap X$ and $t \in S \cap \bar{X}$. Then $(X, \bar{X})$ is a min-cut separating s and t. That is, if $|S| = r$ then we need finding only $\binom{r}{2}$ min-cuts, each separating some $s \in S$ and some $t \in S$. The minimum of these min-cuts gives us $(X, \bar{X})$. The total weight of this cut-set yields θ_1.

We can do better than solving $\binom{r}{2}$ min-cut problems. Gomory and Hu [10] suggest a procedure where one would require solving only $(r - 1)$ min-cut problems, each over a successively reduced network.

7. A HEURISTIC TO ACCELERATE THE CONVERGENCE

One can obtain several nonredundant covers (trees) from a given cover (connected graph) depending upon the sequence in which one attempts to eliminate edges from the cover. Let $x^* = (x_1^*, x_2^*, \cdots, x_m^*)$ be the optimal solution to the linear program associated with SC_t. Intuitively one feels that if $x_i^* < x_j^*$, then there is more likelihood of edge e_j appearing in the optimal solution to SC_t than e_i. Thus a heuristic to obtain a good nonredundant cover is to try to eliminate the edges from the cover in the increasing order of their x values.

From a given nonredundant cover, several A_1^P (matrix of the top k rows of A^P, k being the number of elements in the nonredundant cover) matrices can be generated. The flexibility of this choice is greatly enhanced by the fact that there are so many constraints in the problem. Each A_1^P matrix gives rise to a constraint (if $Q \neq \emptyset$) which has to be appended to the set-covering problem SC_t to obtain SC_{t+1}. Bellmore and Ratliff define a cut $\Sigma_{j \in Q} x_j \geqslant 1$ to be "good" if $\Sigma_{j \in Q} x_j^* < 1$, i.e., the current linear programming optimal solution violates the cut, since this could lead to an improved solution for the linear program associated with SC_{t+1}, and thus give a higher lower bound for the set-covering problem. The following heuristic is developed to get one such "good" cut.

Let us assume, for the sake of simplicity, that $J = \{1, 2, \cdots, k\}$ is a nonredundant cover for SC_t. Recall that, in order to obtain the set $Q = \{j | z_j - c_j > 0\}$ we have to generate the top k rows of the matrix A^P. Suppose we have generated the first w rows, i.e., $a_1, a_2, \cdots, a_w$. Note that for any r, $1 \leqslant r \leqslant w$, $a_{rr} = 1$, $a_{ri} = 0$ for $i \neq r$ and $1 \leqslant i \leqslant k$. Define

$$z_j(w) = \sum_{k=1}^{w} c_k a_{kj}.$$

Clearly, for every j, $z_j = z_j(k) \geqslant z_j(w)$, $w = 1, 2, \cdots, k$. If for some j, $z_j(w) > c_j$, then $z_j > C_j$ and the edge e_j belongs to the set Q. Let $Q(w)$ be the set of edges for which $z_j(w) > c_j$. Let a_{w+1} be any row of the coefficient matrix such that $a_{w+1,w+1} = 1$ and $a_{w+1,j} = 0$, for $1 \leqslant j \leqslant r, j \neq w+1$. Then $e_j \in Q(w+1)$ and $\notin Q(w)$ implies that $a_{w+1,j} = 1$. In an attempt to obtain a good constraint we would like to minimize $\Sigma_{j \in L} x_j^*$, where L is the possible set of edges which belong to set $Q(w+1)$ and not to $Q(w)$. Now

$$\sum_{j \in L} x_j^* = \sum_{j | e_j \in \{E-J-Q(w)\}} a_{w+1,j} x_j^*.$$

Hence the problem is to generate the row a_{w+1} which minimizes the above expression. This is achieved by redefining the weights on edges as

$$c_j^* = \begin{cases} 0 & \text{if } j \in Q(w) \cup \{(e_{w+1})\} \\ \infty & \text{if } j \in J - \{(e_{w+1})\} \\ x_j^* & \text{otherwise.} \end{cases}$$

TABLE I. Summary of computational experience.

n	m	$\lvert S \rvert$	Problems attempted	Problems solved	Avg. CPU time (sec)	Avg. No. iterations	% Improvement over MST
10	20	5	10	10	0.32	2.5	8.9
20	30	10	10	9	3.45	5.6	7.2
20	40	10	10	9	6.76	5.1	8.3
30	40	10	10	7	7.23	6.7	2.5
30	50	15	10	8	15.12	10	12.3
40	50	20	10	6	18.29	12.5	4.7
40	60	20	10	4	27.26	14.2	2.5
50	60	20	10	4	38.73	25	1.2

and obtaining the minimum cut which separates the two nodes of S. If the capacity of this cut is infinity, then the required constraint must exist among the ones added so far.

It should be clear how this heuristic can be modified for problems in Sec. 2.2.

8. COMPUTATIONAL RESULTS AND SUMMARY

The set covering algorithm incorporating the row generation scheme was coded in FORTRAN IV. The linear programming solution technique of Sec. 6 and the heuristic to accelerate the convergence were also used. The sample problems below were solved on IBM 370/158. Each sample problem corresponds to a random connected graph with specified set of nodes and m edges. The procedure for generating the random graphs was as follows: first a random spanning tree over the entire set N of nodes is generated. Additional edges are then added randomly over the network. Random weights between 1 and 10 were then assigned to each edge. The number of nodes in the set S was always chosen to be $n/2$. The table summarizes the computational experience. The last column shows the average percentage improvement obtained when the optimal solution is compared with the weight of the edited minimal spanning tree G_0 of S (a graph on S with edges (s, r) and a minimal length path in G for each edge).

The failure to solve some of these problems was due to the CPU time limit of 60 sec for each problem. Out of 23 problems which did not terminate 20 problems remained at interation 1. This was because the linear program associated with the set covering problem for each of the 20 problems did not terminate during the specified time limit. The linear programming solutions, if obtained, yield very good bounds to the set covering problem. As shown in the table, the edited minimal spanning tree also gives a good bound to the problem. However, there does not seem to be a way of extracting linear programming solutions from this. The experience with this set covering algorithm [6, 7] shows that almost always the termination of the algorithm is due to the linear programming bounds and not due to the condition that $Q = \emptyset$.

For 15 of the 47 problems which were solved, the associated linear programs terminated integer. For 12 more problems the sounded up integer value of the linear program at first iteration was equal to the weight of the Steiner tree obtained using that linear programming solution, leading to the termination of the algorithm. It is interesting to note that in all but 7 problems, the first Steiner tree obtained using LP solutions was later discovered to be the optimal.

The computational results obtained for this problem are in direct contrast to the ones obtained for finding minimal multicommodity disconnecting set [6] where a similar approach yielded very good results. The linear programming solutions obtained there were extremely helpful for fast convergence. In fact the major motivation for this study was computational experience with the disconnecting set problem.

The method, however, presents a new approach to the Steiner problem in graphs. Some method other than linear programming which is able to provide good bounds at each iteration of the algorithm could make this approach much more efficient.

References

[1] R. C. Prim, "Shortest Connection Networks and Some Generalization," *Bell Syst. Tech. J.*, **36**, 1389-1401 (1957).

[2] J. B. Kruskal, Jr., "On the Shortest Spanning Tree of a Graph and the Travelling Salesman Problem," *Proc. Am. Math. Soc.*, **7**, 48-50 (1956).

[3] L. Nastansky, S. M. Selkow, and N. F. Stewart, "Cost Minimal Trees in Directed Acyclic Graphs," *Zeit. Oper. Res.*, **18**, 59-67 (1974).

[4] S. F. Dreyfus and R. A. Wagner, "The Steiner Problem in Graphs," *Networks*, **1**, 195-207 (1972).

[5] S. L. Hakimi, "The Steiner Problem in Graphs," *Networks*, **1**, 113-133 (1971).

[6] Yash P. Aneja, and R. Vemuganti, "A Row Generation Scheme for Finding a Multi-Community Disconnecting Set," *Manag. Sci.*, **23**, 652-659 (1977).

[7] M. Bellmore, and H. D. Ratliff, "Set-Covering and Involutary Bases," *Manag. Sci.*, **18**, 194-206 (1971).

[8] C. E. Lemke, H. M. Salkin, and K. Spielberg, "Set Covering by Single Branch Enumeration with Linear Programming Subproblems," *Oper. Res.*, **19**, 998-1022 (1971).

[9] R. S. Garfinkel and G. L. Nemhauser, *Integer Programming*, Wiley, New York, 1972.

[10] R. E. Gomory, and T. C. Hu., "Multi-Terminal Network Flows," *J. SIAM*, **9** (4), 551-570 (1961).

Received March 1978
Revised March 1979