An Algorithm for the Steiner Problem in Graphs

J. E. Beasley

Department of Management Science, Imperial College, London SW7 2BX, United Kingdom

In this paper we consider the Steiner problem in graphs. This is the problem of connecting together, at minimum cost, a number of vertices in an undirected graph. We present two lower bounds for the problem, these bounds being based on two separate Lagrangian relaxations of a zero–one integer programming formulation of the problem. Problem reduction tests derived from both the original problem and the Lagrangian relaxations are given. Incorporating the bounds and the reduction tests into a tree search procedure enables us to solve problems involving the connection of up to 50 vertices in a graph with 200 undirected arcs and 100 vertices.

I. INTRODUCTION

The Steiner problem in graphs is the problem of connecting together at minimum cost a set of vertices in an undirected graph. The problem is derived from the well-known Euclidean Steiner problem, which is the problem of connecting together points on an Euclidean plane so as to minimize the total length of the edges used. While the Euclidean Steiner problem has been extensively studied, the Steiner problem in graphs has received less attention.

Hakimi [9] and Dreyfus and Wagner [3] were the first to study the problem. Hakimi [9] presented a solution procedure related to an algorithm for the Euclidean Steiner problem due to Melzak [11] but no computational results were given. Dreyfus and Wagner [3] presented an implicit enumeration procedure for the problem, which for a graph containing n vertices (where k of these vertices are to be connected together) requires $O(n^2 2^{k-1})$ operations. Although no computational results were given it is clear that this algorithm is only computationally effective if k is small.

Erickson, Monma, and Veinott [4] have developed a dynamic programming algorithm for the minimum-concave-cost network flow problem which can be applied to the Steiner problem in graphs, leading, as noted by them, to a solution procedure similar to that of Dreyfus and Wagner [3].

Nastansky, Selkow, and Stewart [12] considered the directed form of the problem where there is specified a vertex in the graph and the objective is to find directed paths, of minimum total cost, from this vertex to a given subset of vertices. They presented a

NETWORKS, Vol. 14 (1984) 147–159

CCC 0028-3045/84/010147-13$04.00

tree search procedure based on the linear programming relaxation of a zero-one integer formulation of the problem. Some computational experience was reported.

Foulds and Gibbons [6] presented a tree search procedure based on a bound derived by finding, for each vertex i in the subset of vertices which must be connected together, the minimum cost of connecting that vertex i to some other vertex. Computational experience indicated that only relatively small problems could be solved. Shore, Foulds, and Gibbons [14] later extended this approach and reported the solution of problems involving the connection of up to 25 vertices in a 30-vertex complete graph.

Aneja [1] presented a set-covering formulation of the problem and used a linear programming relaxation, in conjunction with row generation, to tackle the problem. Computational results were given for moderate-sized problems.

Wong [15] has developed a dual-ascent procedure for the problem and presents computational results for the solution of problems involving the connection of up to 30 vertices in a graph containing 120 undirected arcs and 60 vertices.

It is well known that the Steiner problem in graphs is NP-complete [7] and so to solve the problem we will use a tree search procedure based upon the relaxation of a zero-one integer programming formulation of the problem. It is this formulation that we consider in the next section.

II. PROBLEM FORMULATION

In this section we formulate the Steiner problem in graphs as a zero-one integer program, but first we motivate the formulation by making two observations.

Let V be the entire vertex set, K be the set of vertices to be connected together ($K \subseteq V$), E be the set of (undirected) edges $\{(i, j) | i \in V, j \in V\}$ of the graph, and c_{ij} ($\geqslant 0$) be the cost of edge $(i, j) \in E$; then the problem is to choose a subset of E which connects together all the vertices in K at minimum cost.

We can make the following observations: (i) the solution will be a tree, i.e., the edges chosen will form a minimum-cost spanning tree on some set of vertices $K \cup S$ where $S \subseteq V - K$ is called the set of Steiner vertices; (ii) there is a unique elementary path in the solution tree from any vertex $i \in K$ to any other vertex $j \in K$. We can therefore formulate the problem as follows:

Define $K_1 = K - [1]$ (without loss of generality assume $1 \in K$). Let $y_{ij} = 1$ if edge $(i, j) \in E$ is in the solution, 0 otherwise; $x_{ijk} = 1$ if edge $(i, j) \in E$ is on the unique elementary path in the solution tree from 1 to $k \in K_1$, 0 otherwise; then the program is

Minimize

$$\sum_{(i,j)\in E} c_{ij} y_{ij} \tag{1}$$

such that

$$y_{ij} \geqslant x_{ijk}, \quad \forall k \in K_1, \quad \forall (i, j) \in E, \tag{2}$$

$$P_1 \leqslant \sum_{(i,j)\in E} y_{ij} \leqslant P_u, \tag{3}$$

there exists an elementary path (x_{ijk}) from 1 to k, $\forall k \in K_1$, (4)

$$x_{ijk} \in (0, 1), \quad \forall k \in K_1, \quad \forall (i, j) \in E, \tag{5}$$

$$y_{ij} \in (0, 1), \quad \forall (i, j) \in E. \tag{6}$$

Equation (2) ensures that an edge is only on some path in the tree if that edge is in the solution tree and Eq. (3) limits the number of edges in the solution tree. Note that this equation is redundant here but does strengthen the lower bounds we develop in a later section (P_l and P_u are constants where $P_l = |K| - 1$, $P_u = |V| - 1$). Equation (4) ensures that the solution is connected, while Eqs. (5) and (6) are the integrality constraints. Note here that Eq. (4) can be written in a mathematical form (as we shall see below) but we present it in the form given for clarity.

Hence we have a formulation of the Steiner problem in graphs as a zero-one integer program. In the next section we show how we can develop lower bounds for use in a tree search procedure from this formulation of the problem.

III. LOWER BOUNDS

In this section we develop two lower bounds for the Steiner problem in graphs, compare them, and indicate how they can be adapted for use in a tree search procedure for the problem. These bounds are based on two separate Lagrangian relaxations of the formulation of the problem given previously.

A. Lower Bound 1 (LB1)

To develop the first lower bound (LB1) for the problem we relax Eq. (2) in a Lagrangian fashion. Let $s_{ijk} (\geq 0, \forall (i, j) \in E, \forall k \in K_1)$ be the Lagrange multipliers for Eq. (2); then the Lagrangian dual program is given by

Minimize

$$\sum_{(i,j) \in E} \left(c_{ij} - \sum_{k \in K_1} s_{ijk} \right) y_{ij} + \sum_{k \in K_1} \sum_{(i,j) \in E} s_{ijk} x_{ijk} \tag{7}$$

such that

$$P_l \leq \sum_{(i,j) \in E} y_{ij} \leq P_u, \tag{8}$$

there exists an elementary path (x_{ijk}) from 1 to k, $\forall k \in K_1$, (9)

$$x_{ijk} \in (0, 1) \quad \forall k \in K_1, \quad \forall (i, j) \in E, \tag{10}$$

$$y_{ij} \in (0, 1), \quad \forall (i, j) \in E. \tag{11}$$

It is clear that the optimal value of this program, for any set of non-negative Lagrange multipliers, is a lower bound on the optimal solution of the original Steiner problem.

The Lagrangian program can be easily solved as it decomposes into a number of separate problems, the solution consisting of (i) the P_1 y_{ij} with the smallest Lagrangian

objective function [Eq. (7)] coefficients $c_{ij} - \Sigma_{k \in K_1} s_{ijk}$ and then at most $P_u - P_l$ of the remaining y_{ij} providing their Lagrangian coefficients are nonpositive ($\leqslant 0$). (ii) $|K_1|$ shortest elementary path calculations–finding the shortest path from vertex 1 to vertex $k \in K_1$ in the graph with cost matrix (s_{ijk}). These calculations can be easily carried out using, for example, the algorithm of Dijkstra [2].

Let $(Y_{ij}), (X_{ijk})$ represent the optimum values of $(y_{ij}), (x_{ijk})$ in the solution of the Lagrangian program; then the optimal value of the Lagrangian program Z_D is given by

$$Z_D = \sum_{(i,j) \in E} \left(c_{ij} - \sum_{k \in K_1} s_{ijk} \right) Y_{ij} + \sum_{k \in K_1} \sum_{(i,j) \in E} s_{ijk} X_{ijk}. \tag{12}$$

B. Lower Bound 2 (LB2)

To develop the second lower bound (LB2) for the problem we relax Eq. (4) in a Lagrangian fashion. To do this, however, we need to express that equation in a mathematical way. This can be done by redefining the problem as follows: Let A be the set of directed arcs corresponding to the edge set E. Redefine $x_{ijk} = 1$ if arc $(i, j) \in A$ is on the unique elementary directed path in the solution tree from 1 to $k \in K_1$, and 0 otherwise. Then the program becomes

Minimize

$$\sum_{(i,j) \in E} c_{ij} y_{ij} \tag{13}$$

such that

$$y_{ij} \geqslant x_{ijk} + x_{jik} \quad \forall k \in K_1, \ \forall (i, j) \in E, \tag{14}$$

$$P_l \leqslant \sum_{(i,j) \in E} y_{ij} \leqslant P_u, \tag{15}$$

$$\sum_{(1,i) \in A} x_{1ik} \geqslant 1, \quad \forall k \in K_1, \tag{16}$$

$$\sum_{(i,k) \in A} x_{ikk} \geqslant 1, \quad \forall k \in K_1, \tag{17}$$

$$\sum_{(i,j) \in A} x_{ijk} - \sum_{(h,i) \in A} x_{hik} \geqslant 0, \quad \forall k \in K_1, \ \forall i \in V - [1] - [k], \tag{18}$$

$$x_{ijk} \in (0, 1), \quad \forall k \in K_1, \ \forall (i, j) \in A, \tag{19}$$

$$y_{ij} \in (0, 1), \quad \forall (i, j) \in E. \tag{20}$$

This redefinition of the problem, to involve directed arc variables, is necessary as it is difficult to get a convenient formulation of Eq. (4) that preserves the undirected nature of the original problem.

Equations (16)-(18), for a given $k \in K_1$, are the network flow equations for the problem of sending a flow of value 1 from vertex 1 to vertex k, and this network flow problem is equivalent to the problem of finding an elementary path from 1 to k provided the graph has no negative cost circuits. Equation (14) relates the use of an edge

to the use of its directed counterparts. Note here that in this program we implicitly set $x_{kjk} = 0, \forall k \in K_1, \forall (k, j) \in A$ (delete all arcs out of vertex $k \in K_1$) and set $x_{i1k} = 0$, $\forall k \in K_1, \forall (i, 1) \in A$ (delete all arcs into vertex 1).

We relax Eqs. (16)-(18) in a Lagrangian fashion. Let $t_{ik} (\geqslant 0, \forall i \in V - [1] - [k], \forall k \in K_1)$ be the Lagrange multipliers for Eq. (18) and, to ease the notation, let $t_{1k} (\geqslant 0, \forall k \in K_1)$ be the Lagrange multipliers for Eq. (16), and let $t_{kk} (\geqslant 0, \forall k \in K_1)$ be the Lagrange multipliers for Eq. (17).

Then the coefficient of x_{ijk} (C_{ijk}, say) in the Lagrangian dual program is given by

$$\begin{aligned} C_{ijk} &= -t_{ik} - t_{jk}, && i \neq k, \ j = k, \\ &= -t_{ik} + t_{jk}, && i \neq k, \ j \neq 1, k, \\ &= 0, && \text{otherwise,} \end{aligned}$$

and the Lagrangian dual program is:

Minimize

$$\sum_{(i,j) \in E} c_{ij} y_{ij} + \sum_{(i,j) \in A} \sum_{k \in K_1} C_{ijk} x_{ijk} + \sum_{k \in K_1} (t_{1k} + t_{kk}) \tag{21}$$

such that

(14), (15), (19), and (20).

This dual program is easily solved—consider the effect of setting y_{ij} to one, i.e., having edge (i, j) in the solution. From Eq. (14) it is simple to deduce that the best contribution to the dual objective function [Eq. (21)] from y_{ij} (b_{ij}, say) is given by

$$b_{ij} = c_{ij} + \sum_{k \in K_1} \min \{0, C_{ijk}, C_{jik}\} \tag{22}$$

and so the Lagrangian dual program becomes

Minimize

$$\sum_{(i,j) \in E} b_{ij} y_{ij} + \sum_{k \in K_1} (t_{1k} + t_{kk}) \tag{23}$$

such that

$$P_1 \leqslant \sum_{(i,j) \in E} y_{ij} \leqslant P_u, \tag{24}$$

$$y_{ij} \in (0, 1) \qquad \forall (i, j) \in E. \tag{25}$$

The solution to this program consists of the P_1 y_{ij} with the smallest b_{ij} values and then at most $P_u - P_1$ of the remaining y_{ij} provided their Lagrangian coefficients b_{ij} are non-positive ($\leqslant 0$).

As before let $(Y_{ij}), (X_{ijk})$ represent the optimum values of $(y_{ij}), (x_{ijk})$ in the solution of the Lagrangian program; then the optimal value of the Lagrangian program Z_D (a lower bound on the optimal solution of the original Steiner problem) is given by

$$Z_D = \sum_{(i,j)\in E} b_{ij}Y_{ij} + \sum_{k\in K_1} (t_{1k} + t_{kk}). \tag{26}$$

C. Comparison of Bounds

Both lower-bound programs LB1 [Eqs. (7)-(11)] and LB2 [Eqs. (14), (15), (19)-(21)] have the integrality property—that is, the optimal solution value of each program is not altered by removing the integrality restrictions from the constraints of the Lagrangian program (see [8]). This implies that both LB1 and LB2 have the same maximum lower bound value and that this is equal to the optimal value of the linear programming relaxation of the original problem.

LB1 requires $|E|\ |K_1|$ Lagrange multipliers but LB2 requires only $|V|\ |K_1|$ Lagrange multipliers. Hence LB2 gives as tight a bound as LB1 but with fewer Lagrange multipliers. Accordingly we would expect LB2 to give a better computational performance especially as the Lagrangian dual program is easier to solve for LB2 than for LB1.

We present (below) results for both lower bounds so that they can be compared computationally as well as theoretically.

D. Tree Search Adaptation

In adapting the bounds given above for use in a tree search procedure we will, at any stage of the procedure, have a set $F_1(\subseteq E)$ of edges that have been identified as being in the solution tree and a set $F_0(\subseteq E)$ of edges that have been identified as not being in the solution tree. Under these conditions the bounds given above are modified by setting

$$y_{ij} = 1, \quad \forall (i,j)\in F_1 \quad \text{(LB1, LB2)}, \tag{27}$$

$$= 0, \quad \forall (i,j)\in F_0 \quad \text{(LB1, LB2)}, \tag{28}$$

$$s_{ijk} = 0, \quad \forall (i,j)\in F_1, \ \forall k\in K_1 \quad \text{(LB1)} \tag{29}$$

$$= \infty, \quad \forall (i,j)\in F_0, \ \forall k\in K_1 \quad \text{(LB1)}. \tag{30}$$

Note that we also need to modify the solution procedures given before for LB1 and LB2 in the light of these conditions, but these modifications are simple and will not be given here.

IV. THE SUBGRADIENT PROCEDURE

Subgradient optimization was used in an attempt to maximize the lower bounds obtained from the Lagrangian relaxations of the problem. The procedure was as follows:

1. Set initial values for the multipliers of

$$s_{ijk} = 0, \quad \forall (i,j)\in E, \ \forall k\in K_1 \quad \text{(LB1)}, \tag{31}$$

$$t_{ik} = 0, \quad \forall i\in V, \ \forall k\in K_1 \quad \text{(LB2)}. \tag{32}$$

2. Solve the Lagrangian dual program with the current set of multipliers and let the solution be $Z_D, (Y_{ij}), (X_{ijk})$.
3. If the Lagrangian solution (Y_{ij}) is a feasible solution to the original problem then update Z_{UB}, the upper bound on the problem corresponding to a feasible solution, accordingly. Update the maximum lower bound found (Z_{max}) with Z_D.
4. Stop if $Z_{UB} = Z_{max}$ since then Z_{UB} is the optimal solution, else go to Step 5.
5. Calculate the subgradients

$$G_{ijk} = X_{ijk} - Y_{ij}, \quad \forall (i,j) \in E, \quad \forall k \in K_1 \quad \text{(LB1)}, \tag{33}$$

$$H_{1k} = 1 - \sum_{(1,i)\in A} X_{1ik}, \quad \forall k \in K_1 \quad \text{(LB2)}, \tag{34}$$

$$H_{kk} = 1 - \sum_{(i,k)\in A} X_{ikk}, \quad \forall k \in K_1 \quad \text{(LB2)}, \tag{35}$$

$$H_{ik} = \sum_{(h,i)\in A} X_{hik} - \sum_{(i,j)\in A} X_{ijk}, \quad \forall k \in K_1, \ \forall i \in V - [1] - [k] \quad \text{(LB2)}. \tag{36}$$

6. Define a step size T by

$$T = f(Z_{UB} - Z_D) \Big/ \left(\sum_{k\in K_1} \sum_{(i,j)\in E} (G_{ijk})^2 \right) \quad \text{(LB1)}, \tag{37}$$

$$T = f(Z_{UB} - Z_D) \Big/ \left(\sum_{i\in V} \sum_{k\in K_1} (H_{ik})^2 \right) \quad \text{(LB2)}, \tag{38}$$

where $0 < f \leqslant 2$ and update the Lagrange multipliers by

$$s_{ijk} = \max \{0, s_{ijk} + TG_{ijk}\}, \quad \forall (i,j) \in E, \quad \forall k \in K_1 \quad \text{(LB1)}, \tag{39}$$

$$t_{ik} = \max \{0, t_{ik} + TH_{ik}\}, \quad \forall i \in V, \quad \forall k \in K_1 \quad \text{(LB2)}. \tag{40}$$

7. Go to Step 2 to resolve the Lagrangian program with this new set of multipliers unless sufficient subgradient iterations have been performed, in which case stop.

In calculating a value for f [Eqs. (37), (38)] we followed the approach of Held, Wolfe, and Crowder [10] in letting $f = 2$ for $|V|$ iterations, then successively halving both f and the number of iterations until the number of iterations reached a threshold value of 5; f was then halved every 5 iterations. The subgradient procedure was stopped when f fell below 0.05.

V. PROBLEM REDUCTION

Various tests can be carried out that will reduce the size of the problem (in terms of vertices/edges) that we need consider. In this section we outline those that we found most effective.

As before let F_1 $(\subseteq E)$ be the set of edges that have been identified as being in the solution tree and let F_0 $(\subseteq E)$ be the set of edges that have been identified as not being in the solution tree. Let d_{ij} represent the least cost of an elementary path from $i \in V$

to $j \in V$ in the graph with edge costs given by

$$\begin{cases} c_{ij} & \text{if } (i,j) \in E - F_1 - F_0, \\ 0 & \text{if } (i,j) \in F_1, \\ \infty & \text{otherwise.} \end{cases}$$

Note that d_{ij} can be viewed as the incremental cost of a path from i to j having regard to the edges already identified as being in the solution tree. The matrix (d_{ij}) can be easily calculated by use of Floyd's [5] algorithm. We can now outline the reduction tests that we found effective.

A. Least Cost

Any edge $(i, j) \in E - F_1 - F_0$ for which $d_{ij} < c_{ij}$ can be eliminated from the problem.

B. Degree Tests

Let D_i be the degree of a vertex $i \in V$ as calculated from the edge set $E - F_0$; then any vertex $i \in V - K$ for which $D_i \leqslant 1$ can be eliminated from the problem. Similarly, for any vertex $i \in V - K$ for which $D_i = 2$ we have that if $p \in V$, $q \in V$ are the vertices to which i is joined and $(i, p) \notin F_1$, $(i, q) \notin F_1$ then (i) if $d_{pq} < c_{pi} + c_{iq}$ then we eliminate i from the problem; (ii) if $d_{pq} = c_{pi} + c_{iq}$ then we eliminate i from the problem, add a new edge (p, q) of cost $c_{pi} + c_{iq}$ to E [note that if there is already an edge (p, q) then we assign it a cost of min $\{c_{pq}, c_{pi} + c_{iq}\}$], and decrease P_1 (the minimum number of edges in the solution tree) by one.

C. Nearest Vertex

For any vertex $k \in K$ let $i \in V$ be the vertex such that $c_{ki} = \min_{(k,j) \in E - F_0} c_{kj}$; i.e., i is the nearest vertex to k. Then if

$$c_{ki} + \min_{\substack{j \in K \\ j \neq k}} (d_{ij}) \leqslant \min_{\substack{(k,j) \in E - F_0 \\ j \neq i}} (c_{kj}), \tag{41}$$

we can set $y_{ki} = 1$ [i.e., edge (k, i) is in the solution tree]. The left-hand side of Eq. (41) is the worst cost we will incur in connecting $k \in K$ to some other vertex in K if we use the edge (k, i), while the right-hand side of Eq. (41) is the minimum cost we will incur if we do not use edge (k, i). Note here that care must be taken in applying this test to ensure that in the event of equality in Eq. (41) we do not join vertex $k \in K$ to two (or more) vertices all an equal cost away from vertex k. This reduction test can be viewed as a generalization of the Prim [13] algorithm for the shortest spanning tree of a graph and when $K = V$ is the same as the Prim algorithm.

D. Reachability

If for any vertex $i \in V - K$ we have that

$$\sum_{(i,j) \in F_1} c_{ij} + \max_{k \in K_1} d_{ik} > Z_{UB} \tag{42}$$

then vertex i can be eliminated from the problem (as to include it in the solution tree would force the solution value above the least-cost solution Z_{UB} already found).

E. Penalties on Number of Edges

It is clear from the structure of the Lagrangian lower-bound programs LB1 and LB2 that we can calculate a lower bound on the solution obtained with exactly m edges in the solution tree. By investigating all values of m where $P_l \leqslant m \leqslant P_u$ and comparing the lower bounds obtained with Z_{UB}, the upper bound on the problem corresponding to a feasible solution, we can update P_l and P_u accordingly (e.g., if the lower bound when $m = P_l$ is greater than Z_{UB} then we can increase P_l by one).

F. Edge Penalties

It is also clear from the Lagrangian programs that penalties can be calculated for the setting of y_{ij} to one (or zero), i.e., having edge (i, j) in the solution tree (or not). For example, the lower bound obtained when setting $y_{ij} = 1$ (where $Y_{ij} = 0$) is given (for LB2) by

$$Z_D + b_{ij} - \max_{\substack{(p,q)\in E \\ Y_{pq}=1}} (b_{pq}), \quad \text{if} \sum_{(p,q)\in E} Y_{pq} = P_u, \tag{43}$$

$$Z_D + b_{ij} - \max \left\{ 0, \max_{\substack{(p,q)\in E \\ Y_{pq}=1}} (b_{pq}) \right\}, \quad \text{otherwise.} \tag{44}$$

If this lower-bound penalty exceeds Z_{UB} then we know that edge (i, j) can never be in the solution and we can remove it from the edge set E. Expressions similar to (43) and (44) apply for setting $y_{ij} = 0$ (when $Y_{ij} = 1$), for both LB1 and LB2.

G. Components

The set F_1 of edges that have been identified as being in the solution tree will define a number of connected components (each component consisting of vertices connected by edges in F_1 in a tree structure). We can use such components to reduce the size of the problem. Suppose we have two vertices p, q (say) in the same component with $p \in K$ and $q \in K$. Then essentially the constraints [Eq. (4)] in the formulation of the problem relating to p and q are the same since p and q are joined (so any path to p means there is a path to q and vice versa). Hence we can neglect one of these constraints (say q) by removing q from K, i.e., setting

$$s_{ijq} = 0, \quad \forall (i, j) \in E \quad \text{(LB1)}, \tag{45}$$

$$X_{ijq} = 0, \quad \forall (i, j) \in A \quad \text{(LB2)}, \tag{46}$$

$$t_{iq} = 0, \quad \forall i \in V \quad \text{(LB2)}, \tag{47}$$

So for each component we need at most one active vertex $p \in K$ for which we require a path from 1 to p. Note that for any component involving vertex 1 we need no active vertex $p \in K$ since a path from 1 to p already exists.

VI. THE TREE SEARCH PROCEDURE

We used a binary depth-first tree search procedure computing at each tree node a lower bound for the optimal completion of the node. We indicate below how we structured the initial tree node and the tree search nodes with respect to the calculation of the bound and the reduction tests we carried out.

A. Initial Tree Node

1. Upper Bound

At the initial tree node we first chose the vertex to act as the root vertex (denoted by $1 \in K$ in this paper). This was the vertex $j \in K$ where

$$\sum_{k \in V} d_{jk} \geqslant \sum_{k \in V} d_{ik}, \quad \forall i \in K; \tag{48}$$

i.e., the root vertex was the vertex "furthest" from all the other vertices in K (ties broken arbitrarily).

We then found an initial feasible solution Z_{UB} from the edited minimal spanning tree on the graph with edge set E and edge cost (d_{ij}) (as discussed by Aneja [1]).

2. Reduction

The reduction tests outlined in the previous section relating to least cost, nearest vertex, degree, reachability, and components were then all performed until no further reduction could be achieved.

3. Lower Bound

The subgradient procedure was then carried out with at each subgradient iteration the penalty tests relating to the number of edges in the solution and for the setting of y_{ij} to one (or zero) being carried out. At the end of the subgradient procedure the set of multipliers associated with the best lower bound found were recalled and the Lagrangian dual program resolved with that set of multipliers.

4. Reduction

All the reduction tests outlined in Section V were then performed until no further reduction was possible.

B. Other Tree Nodes

1. Reduction

All the reduction tests outlined in Section V were carried out at each tree node in the same manner as described in Sections VI.A.2–VI.A.4 above for the initial tree node. The matrix (d_{ij}) is computationally expensive to calculate, involving $O(|V|^3)$ operations initially, but as the algorithm proceeds the components that are formed enable the calculation of (d_{ij}) to be considerably speeded [since each component can be re-

garded as a single "vertex" for the calculation of (d_{ij})]. Consequently we calculated the matrix (d_{ij}) at each tree node.

2. Bound

At each tree node we carried out five subgradient iterations with $f = 1$ (these parameters being decided upon after exploring a number of possibilities). The initial set of multipliers at each tree node were the set associated with the best lower bound found at the predecessor tree node.

3. Branching

In forward branching we branched by choosing a vertex $p \in V - K$ to specify as being in the solution tree. Computationally this can be achieved by adding p to the set K and setting

$$s_{ijp} = 0, \quad \forall (i, j) \in E \quad \text{(LB1)}, \tag{49}$$

$$t_{ip} = 0, \quad \forall i \in V \quad \text{(LB2)}. \tag{50}$$

Note here that often adding a vertex to K enables further reduction of the problem to be achieved by the nearest vertex reduction test. We chose the vertex $p \in V - K$ to be such that y_{pq} has the smallest Lagrangian coefficient in the Lagrangian objective function [Eq. (7), LB1, and Eq. (23), LB2] for some $q \in V$, ties for p broken arbitrarily.

4. Backtracking

We can backtrack in the tree when $Z_D > Z_{UB}$ or when the reduction tests indicate that there is no feasible completion of the current tree node (e.g., as may happen if the graph becomes disconnected).

VII. COMPUTATIONAL RESULTS

The algorithms LB1 and LB2 were programmed in FORTRAN and run on a CDC 7600 using the FTN compiler with maximum optimization for a number of randomly generated problems.

All the problems generated were produced using the scheme outlined by Aneja [1]. For a given $|V|$ we produced six problems corresponding to $|K| = \frac{1}{6}|V|, \frac{1}{4}|V|, \frac{1}{2}|V|$, and average vertex degrees of 2.5 and 4.

Table I gives the results of the algorithms for the problems attempted. In that table we show, for each problem, the duality gap at the end of the subgradient procedure at the initial tree node where the duality gap is measured by (optimal value - maximum lower bound at initial tree node)/(optimal value) X 100%. (Note here that for the problems in which the tree search was not completed the duality gap was calculated using the best upper bound found.)

We note that LB1 gives a lower duality gap for all the problems we attempted and, in general, LB2 takes less time than LB1, although for only a few problems [(6), (10), (12), (15)] is the difference in time between LB2 and LB1 particularly marked.

TABLE I. *Computational results.*

Problem number	$\|V\|$	$\|E\|$	$\|K\|$	Bound	Initial tree node: Number of subgradient iterations	Initial tree node: Duality gap (%)	Initial tree node: After all reductions $\|V\|$	Initial tree node: After all reductions $\|E\|$	Initial tree node: After all reductions $\|F_1\|$	Initial tree node: CDC 7600 time (s)	Tree search: Number of tree nodes	CDC 7600 total time[a] (s)
1	50	63	9	LB1/LB2	...	...	...	...	...	...	...	1.0
2			13	LB1/LB2	...	...	...	...	...	...	...	1.0
3			25	LB1/LB2	...	...	...	...	...	...	...	1.0
4	50	100	9	LB1	103	10.42	31	57	6	3.9	13	6.1
				LB2		17.54	31	57	6	3.3	11	5.5
5			13	LB1		19.95	37	71	7	5.8	3	7.1
				LB2		37.07	37	71	7	4.5	9	6.5
6			25	LB1		15.84	44	70	21	7.0	39	15.1
				LB2		21.75	44	70	21	5.3	31	11.6
7	75	94	13	LB1/LB2	...	...	...	...	...	...	...	3.0
8			19	LB1/LB2	...	...	...	...	...	...	...	2.2
9			38	LB1/LB2	...	...	...	...	...	...	...	2.3
10	75	150	13	LB1	149	16.63	53	109	7	15.0		(250)
				LB2		30.91	53	109	7	11.2	245	124.4
11			19	LB1		8.93	49	81	20	8.2	9	11.6
				LB2		12.39	52	89	20	6.5	7	9.1
12			38	LB1		12.00	59	90	34	12.9	53	29.5
				LB2		17.94	59	90	34	12.0	23	19.1
13	100	125	17	LB1	198	14.38	44	63	16	13.6	37	28.4
				LB2		25.73	44	63	16	11.9	57	31.0
14			25	LB1		7.28	52	67	33	13.1	49	26.4
				LB2		14.92	52	67	33	12.7	37	24.1
15			50	LB1		5.48	66	77	51	13.9	11	19.0
				LB2		9.75	66	77	51	14.7	29	24.2
16	100	200	17	LB1	198	17.32	76	155	15	29.5		(250)
				LB2		29.55	76	155	15	23.7		(250)
17			25	LB1		4.40	57	83	32	16.3	7	20.9
				LB2		12.64	57	84	32	14.7	5	18.9
18			50	LB1		5.08	73	86	56	14.5	7	19.5
				LB2		8.94	73	86	56	15.0	9	20.6

[a]A time in brackets indicates that the algorithm did not finish in the time shown.

VIII. CONCLUSIONS

In this paper we have considered the Steiner problem in graphs and developed a tree search procedure for solving the problem based upon Lagrangian relaxations of a zero-one integer programming formulation of the problem. Computational results indicate that the algorithm is an effective procedure for large problems.

References

[1] Y. P. Aneja, An integer linear programming approach to the Steiner problem in graphs. *Networks* 10 (1980) 167–178.

[2] E. W. Dijkstra, A note on two problems in connection with graphs. *Numer. Math.* 1 (1959) 269.

[3] S. E. Dreyfus and R. A. Wagner, The Steiner problem in graphs. *Networks* 1 (1972) 195–207.

[4] R. E. Erickson, C. L. Monma, and A. F. Veinott, Minimum-concave-cost network flows. Technical Report, Bell Labs., Holmdel, NJ (1979).

[5] R. W. Floyd, Algorithm 97–shortest path. *Comm. ACM.* 5 (1962) 345.

[6] L. R. Foulds and P. B. Gibbons, A branch and bound approach to the Steiner problem in graphs. In *Proceedings of the 14th Annual Conference of the Operational Research Society of New Zealand*, University of Canterbury, Christchurch, New Zealand (1978), *Vol. 1–Theory*, pp. 61–70.

[7] M. R. Garey and D. S. Johnson, Computers and intractability: A guide to the theory of NP-completeness. Freeman, San Francisco (1979), pp. 208, 209.

[8] A. M. Geoffrion, Lagrangian relaxation and its uses in mathematical programming. *Math. Programming Study* 2 (1974) 82–114.

[9] S. L. Hakimi, Steiner's problem in graphs and its implications. *Networks* 1 (1971) 113–133.

[10] M. Held, P. Wolfe, and H. P. Crowder, Validation of subgradient optimisation. *Math. Programming* 6 (1974) 62–88.

[11] Z. A. Melzak, On the problem of Steiner. *Canad. Math. Bull.* 4 (1961) 143–148.

[12] L. Nastansky, S. M. Selkow, and N. F. Stewart, Cost-minimal trees in directed acyclic graphs. *Z. Operations Res.* 18 (1974) 59–67.

[13] R. C. Prim, Shortest connection networks and some generalizations. *Bell System Tech. J.* 36 (1957) 1389.

[14] M. L. Shore, L. R. Foulds, and P. B. Gibbons, An algorithm for the Steiner problem in graphs. *Networks* 12 (1982) 323–333.

[15] R. T. Wong, A dual ascent approach for Steiner tree problems on a directed graph. To appear *Math. Programming.*

Received July 22, 1982
Accepted July 21, 1983