A Branch and Cut Algorithm for the Steiner Problem in Graphs

A. Lucena,[1] J. E. Beasley[2]

[1] Laboratorio Nacional de Computacao Cientifica/CNPq, Rua Lauro Muller 455, Rio de Janeiro RJ 22290-060, Brazil

[2] The Management School, Imperial College, London SW7 2AZ, England

Received 5 August 1992; accepted 4 August 1997

Abstract: In this paper, we consider the Steiner problem in graphs, which is the problem of connecting together, at minimum cost, a number of vertices in an undirected graph with nonnegative edge costs. We use the formulation of this problem as a shortest spanning tree (SST) problem with additional constraints given previously in the literature. We strengthen this SST formulation and present a branch and cut algorithm to solve the problem to optimality. This algorithm incorporates reduction tests and is used to solve a number of problems drawn from the literature. A number of general issues relating to branch and cut algorithms are also highlighted. *© 1998 John Wiley & Sons, Inc.* Networks **31:** 39–59, 1998

Keywords: Steiner problem; branch and cut

1. INTRODUCTION

The Steiner problem in graphs (henceforth, SPG) is the problem of connecting together, at minimum cost, a set of vertices in an undirected graph. In a previous paper, Beasley [4] introduced a formulation of the SPG as a shortest spanning tree (SST) problem with additional constraints. In this paper, we derive a lower bound for the SPG based upon a linear programming (LP) relaxation of a strengthened version of this SST formulation of the problem. This procedure for generating a lower bound leads naturally to a branch and cut tree search procedure for optimally solving the problem.

1.1. Literature Survey

In a comprehensive survey paper [28], the literature relating to the SPG was reviewed and so, for reasons of space, we shall only deal here with papers additional to those discussed in [28]. We should also mention here the book by Hwang et al. [29] relating to the Steiner problem.

Berman and Ramaiyer [7] presented an approximation algorithm with a known worst-case ratio based upon vertex restricted edge-disjoint full Steiner trees. Chopra and Rao [10, 11] presented papers studying both undirected and directed (replace each undirected edge by two directed edges) formulations of the SPG. They showed [10] that the LP relaxation of the directed formulation is stronger than the LP relaxation of the undirected formulation. Facet-defining inequalities were also presented. No computational results were given.

Chopra et al. [9] presented a branch and cut algorithm based upon the directed formulation of the problem suggested in Chopra and Rao [10, 11]. Cutting planes associated with cut-sets separating vertices that must be in the solution tree were used. Extensive computational results were given.

Correspondence to: J. E. Beasley

Chopra and Gorres [8] considered the node-weighted SPG and applied a similar approach to that given in [9]. Dowsland [14] presented an algorithm for the SPG based upon a local optimization heuristic and simulated annealing. Computational results were given for problems with up to 100 vertices and 4950 edges.

Duin [15], in a PhD thesis, presented a number of new results for the SPG. In particular, reduction tests were presented that, computationally, appear very effective. Incorporating these reduction tests into a tree search procedure, using a lower bound due to Wong [57], enables him to solve to optimality problems involving up to 1000 vertices and 25,000 edges in impressive computational times.

Duin and Voß [19] presented a heuristic for the SPG based upon vertex and edge exchanges and presented computational results for problems involving up to 1000 vertices. Esbensen [20] presented a genetic algorithm for the SPG. He used problem reduction tests and presented computational results for problems involving up to 2500 vertices and 62,500 edges. Floren [25] presented a simplification of the algorithm due to Mehlhorn [40].

Goemans and Myung [26] presented some formulations of the SPG and showed that a number of these are equivalent. No computational results were given. Kapsalis et al. [30] presented a genetic algorithm for the SPG. Computational results were given for problems involving up to 100 vertices and 200 edges.

Khoury and Pardalos [31] presented a heuristic for the SPG based upon Prim's [48] algorithm for the minimal spanning tree. Computational results were given for a number of problems involving up to 500 vertices and 2500 edges. Khoury et al. [32] presented a procedure for generating nontrivial test problems for the SPG that have known optimal solutions. Khoury et al. [33] presented a number of formulations of the SPG and of the directed version of the problem.

Lucena [35] applied a number of the ideas presented in this paper, but in the context of Lagrangean relaxation, rather than branch and cut. Computational results were given for problems involving up to 2500 vertices and 62,500 edges. Lucena [36] also presented an algorithm for the SPG incorporating Lagrangean relaxation, Lagrange cuts, and linear programming. Computational results were given for problems involving up to 2500 vertices and 62,500 edges. Additionally, Lucena [37] presented an algorithm for the SPG improving on the preliminary results presented in [35]. A significant difference between the work presented in [37] and the work presented in this paper was his use of generalized subtour elimination constraints. Computational results were given for problems involving up to 2500 vertices and 62,500 edges.

Matsui and Yabe [39] presented a lower bound for the SPG based upon an edge-covering problem. Computational results were given for problems with up to 100 vertices and 4950 edges. Plesnik [46] presented an improved analysis of the worst-case performance of his earlier contraction heuristic [45] together with a revised contraction heuristic. He also presented an analysis comparing standard heuristics that have been presented in the literature.

Pornavalai et al. [47] presented a Hopfield neural network model for the SPG. Computational results were presented for problems involving up to 14 vertices. Verhoeven et al. [51] presented a local search heuristic for the SPG based upon a neighborhood involving edge exchanges and presented computational results for large-sized problems.

Voß [52] presented a classification of a large number of heuristics for the SPG. Extensive computational results with regard to the quality of solution obtained by these heuristics were given. Voß [53] also presented details of a number of special cases of the SPG which are solvable in polynomial time (e.g., if the underlying graph is series-parallel). He posed the question as to whether, for SPG with degree constraints, there are also special cases solvable in polynomial time. Voss [54] presented an analysis of the worst-case performance of a number of heuristics for the directed version of SPG and concluded that no one heuristic dominated the others with respect to worst-case performance.

Wade and Rayward-Smith [55] presented a number of heuristics for the SPG based upon simulated annealing and presented computational results for problems involving up to 2500 vertices and 62,500 edges. Winter and Smith [56] presented an integrative overview of a number of heuristics for the SPG. Extensive computational results were given.

2. PROBLEM FORMULATION

In this section, we first formulate the SPG as a restricted SST problem (as in [4]) and then strengthen that formulation.

2.1. Restricted SST Formulation

Let

V	be the entire vertex set
K	be the set of vertices which are to be connected together ($K \subseteq V$) and, without loss of generality, assume that $1 \in K$
E	be the set of (undirected) edges $\{(i, j) \mid i < j,\ i \in V, j \in V\}$ of the graph
c_{ij}	be the cost of edge $(i, j) \in E (c_{ij} > 0)$.

Then, the SPG is to choose a subset of E which connects together all the vertices in K at minimum cost. The solution to this problem constitutes a minimum-cost spanning tree (called the Steiner tree) on some set of vertices $K \cup S$, where $S \subseteq V - K$ is called the set of Steiner vertices.

Beasley [4] showed that this problem can be regarded as equivalent to the problem of finding the SST, subject to additional constraints, on a related graph. Specifically,

(a) add an artificial vertex (vertex 0) to the graph;
(b) for each vertex $i \in V - K$, add an edge $(0, i)$ of cost zero;
(c) for vertex $1 \in K$, add an edge $(0, 1)$ of cost zero; and
(d) find the SST of the resulting graph subject to the additional restriction that in that SST any vertex $i \in V - K$ connected by the edge $(0, i)$ to vertex 0 must have degree one.

In [4], it is shown that the above restricted SST formulation, defined with respect to the graph involving the artificial vertex, can be regarded as equivalent to the original SPG on the graph (V, E). In particular, they have the same optimal solution value and a similar solution structure.

Figure 1 (from [4]) illustrates the optimal solution to the restricted SST problem. Note from that figure that vertices not in the Steiner tree are directly connected only to vertex 0 while the Steiner tree is based around vertex 1 ($\in K$).

To formulate the restricted SST problem mathematically, let

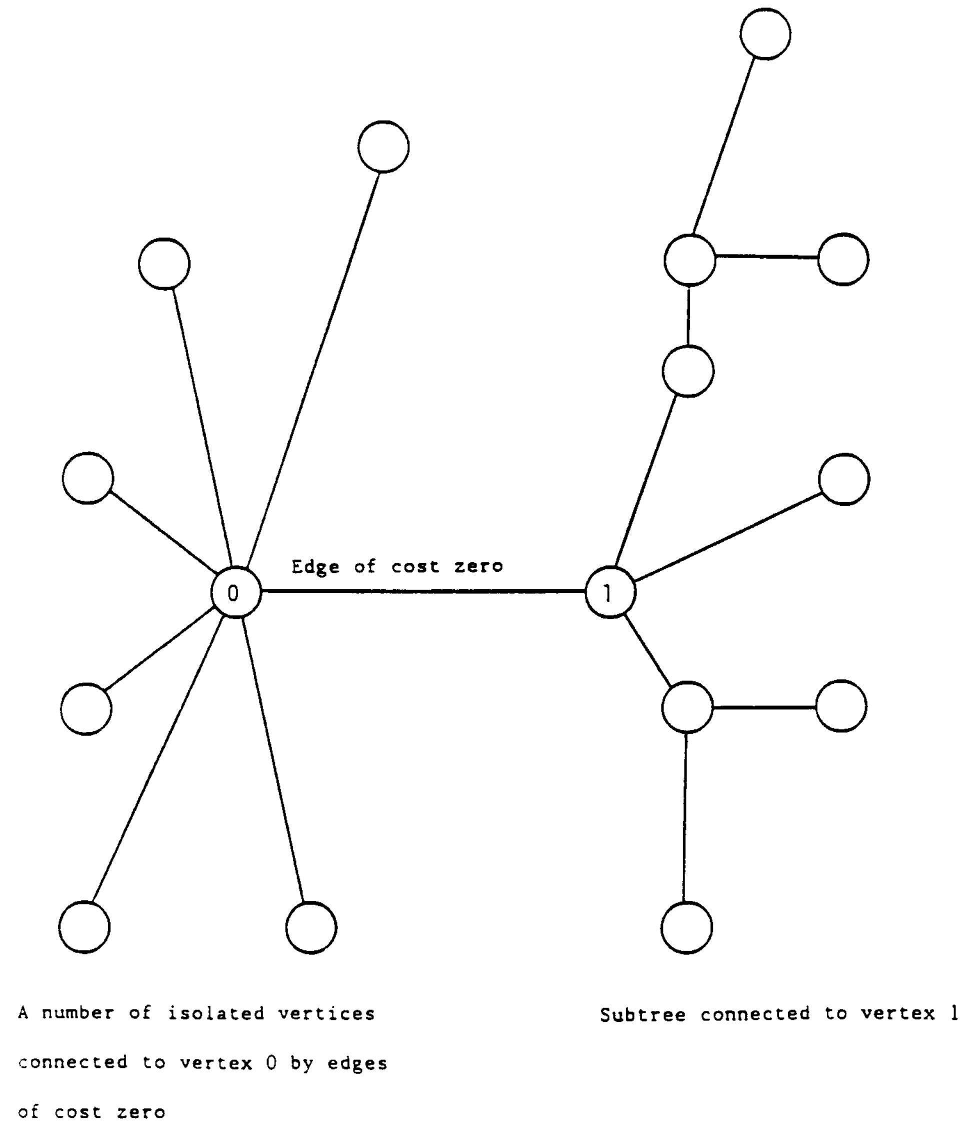
Edge of cost zero
0
1
A number of isolated vertices connected to vertex 0 by edges of cost zero
Subtree connected to vertex 1

Fig. 1.

V_0 represent $V \cup \{0\}$,
E_0 represent $E \cup \{(0, i) | i \in V - K + \{1\}\}$, and
P_i represent $\{(i, j) | (i, j) \in E\} \cup \{(j, i) | (j, i) \in E\}$ so that P_i is the set of edges of E which involve vertex i.

Defining,

$c_{0i} = 0 \ \forall i \in V - K + \{1\}$
$x_{ij} = 1$ if edge $(i, j) \in E_0$ is in the optimal solution
$= 0$ otherwise.

The formulation of SPG as a restricted SST problem is

$$\text{minimize} \quad \sum_{(i,j) \in E_0} c_{ij} x_{ij} \tag{1}$$

$$\text{subject to} \quad \sum_{(p,q) \in E_0} x_{pq} = |V_0| - 1 \tag{2}$$

$$\sum_{p,q \in T \ (p,q) \in E_0} x_{pq} \leq |T| - 1 \quad \forall T \subseteq V_0 \tag{3}$$

$$x_{0i} + x_{pq} \leq 1 \quad \forall (p, q) \in P_i \quad \forall i \in V - K \tag{4}$$

$$x_{ij} \in (0, 1) \quad \forall (i, j) \in E_0. \tag{5}$$

Constraints (2) and (3) ensure that the edges chosen form a spanning tree on the graph with vertex set V_0 and edge set E_0 [constraint (2) relates to the number of edges used and constraint (3) relates to subtour elimination]. The exclusion constraint (4) ensures that if edge $(0, i)$ is used $(i \in V - K)$ no other edge which involves vertex i can be used (and, hence, that vertex i is directly connected only to vertex 0, implying that it is not in the Steiner tree—see Fig. 1). Constraint (5) is the integrality constraint.

2.2. Stronger Formulation

To obtain a tighter LP relaxation than that otherwise derived from Eqs. (1)–(5), we strengthen the formulation by introducing degree constraints, a connectivity constraint, and a constraint ensuring edge (0, 1) is in the solution. As a result, a better approximation of the convex hull of integer solutions is obtained from the LP relaxation of the problem. We deal with each of these constraints in turn.

(1) Degree Constraints

In the optimal solution to the SPG, any Steiner vertex must have a vertex degree of at least two (since, if not, it would have degree one, whereupon the vertex, and the edge connecting it to the rest of the optimal Steiner tree, could be removed—contradicting the optimality assumption). This constraint can be expressed (e.g., see [16]) as

$$\sum_{(p,q) \in P_i} x_{pq} \geq 2(1 - x_{0i}) \quad \forall i \in V - K \tag{6}$$

(since if $x_{0i} = 0$, vertex $i \in V - K$ is in the Steiner tree and constraint (6) ensures that at least two edges involving vertex i are used; if $x_{0i} = 1$, then constraint (6) is redundant).

(2) Connectivity Constraints

In any feasible solution to the SPG there must be a path from vertex $1 \in K$ to all other vertices $k \in K - \{1\}$ (for the solution to be connected). This constraint can be expressed as

$$\sum_{p \in W \ q \in V_0 - W \ (p,q) \in E_0} x_{pq} \geq 1 \quad 1 \in W \quad k \in V_0 - W \quad \forall W \subseteq V_0 \quad \forall k \in K - \{1\}, \tag{7}$$

i.e., there must be an edge connecting any subset $W (\subseteq V_0)$ of vertices (with $1 \in W$) to vertices outside the subset. This constraint was used in [1] as well as in [3, 9].

(3) Edge (0, 1)

Since edge (0, 1) must appear in the optimal solution, we can add to the formulation the constraint

$$x_{01} = 1. \tag{8}$$

(4) Strengthened Formulation

Hence, the complete strengthened formulation of the problem is

$$\text{minimize} \quad \sum_{(i,j) \in E_0} c_{ij} x_{ij} \tag{9}$$

$$\text{subject to} \quad \sum_{(p,q) \in P_i} x_{pq} \geq 2(1 - x_{0i}) \quad \forall i \in V - K \tag{10}$$

$$x_{01} = 1 \tag{11}$$

$$\sum_{p \in W \ q \in V_0 - W \ (p,q) \in E_0} x_{pq} \geq 1 \quad 1 \in W \quad k \in V_0 - W \quad \forall W \subseteq V_0 \quad \forall k \in K - \{1\} \tag{12}$$

$$\sum_{(p,q) \in E_0} x_{pq} = |V_0| - 1 \tag{13}$$

$$\sum_{p,q \in T \ (p,q) \in E_0} x_{pq} \leq |T| - 1 \quad \forall T \subseteq V_0 \tag{14}$$

$$x_{0i} + x_{pq} \leq 1 \quad \forall (p, q) \in P_i \quad \forall i \in V - K \quad (15)$$

$$x_{ij} \in (0, 1) \quad \forall (i, j) \in E_0. \quad (16)$$

This formulation of the problem has $|E_0|$ variables, $0(|V - K| + |E|)$ explicit constraints, and an exponential number of connectivity and subtour elimination constraints.

3. LP LOWER BOUND

In this section, we derive an initial LP lower bound for the SPG by neglecting all exclusion, connectivity, and subtour elimination constraints [constraints (15), (12), and (14), respectively]. We then discuss how we solve the separation problem (see Nemhauser and Wolsey [41]) for these constraints.

3.1. Initial LP

To generate an LP lower bound for SPG from our strengthened formulation [Eqs. (9)–(16)], we obviously need to replace the integrality constraint (16) by the corresponding linear constraint. We also chose to initially neglect *all* exclusion, connectivity, and subtour elimination constraints.

Therefore, very much in accordance with branch and cut algorithms in the literature (e.g., Padberg and Rinaldi [43]), our initial LP relaxation of the problem is

$$\text{minimize} \quad \sum_{(i,j)\in E_0} c_{ij}x_{ij} \quad (17)$$

$$\text{subject to} \quad \sum_{(p,q)\in P_i} x_{pq} \geq 2(1 - x_{0i}) \quad \forall i \in V - K \quad (18)$$

$$x_{01} = 1 \quad (19)$$

$$\sum_{(p,q)\in E_0} x_{pq} = |V_0| - 1 \quad (20)$$

$$0 \leq x_{ij} \leq 1 \quad \forall (i, j) \in E_0. \quad (21)$$

This LP has only $|E_0|$ variables and $(|V - K| + 2)$ constraints and so, computationally, should be relatively easy to solve.

3.2. Separation Problem

Let the solution to the above LP [Eqs. (17)–(21)] be given by $\{X_{ij}\}$ with value Z_{LB} (a lower bound on the optimal solution to the original SPG). The likelihood is that this LP solution does not satisfy all the constraints that we have initially neglected [constraints (12), (14), and (15)]. Constraints which are not satisfied by the current LP solution $\{X_{ij}\}$, but which are valid constraints for the original problem (SPG), are called violated constraints.

The separation problem is to identify violated constraints: exclusion, connectivity, or subtour elimination constraints (or to determine that no such violated constraints exist). Any violated constraints which can be identified can be added to the LP to (hopefully) strengthen the lower bound (Z_{LB}) obtained from that LP.

(1) Exclusion Constraints

It is clear that any exclusion constraints (15) which are violated can be easily identified (namely, in linear time by inspection).

(2) Connectivity Constraints

Violated connectivity constraints (12) can be easily identified by, for each $k \in K - \{1\}$,

(a) finding the maximum flow from vertex 1 ($\in K$) to k in a graph with edge capacities $\{X_{ij}\}$; and
(b) examining the capacity of the cutset associated with that maximum flow.

If the capacity of this cutset is less than 1, then the cutset defines a violated connectivity constraint.

(3) Subtour Elimination Constraints

Identifying violated subtour elimination constraints (14) is computationally more demanding than identifying violated exclusion or connectivity constraints. Padberg and Wolsey [44] showed that, given an LP solution $\{X_{ij}\}$, it is possible to find a subtour elimination constraint that is violated by this solution (or to prove that there is no such violated constraint) by solving a sequence of maximum flow problems in an appropriately defined directed network.

Padberg and Wolsey [44] showed that, at most, $|V_0| - 2$ maximum flow problems need to be solved in a directed network which has (essentially), at most, $(|V_0| + 2)$ vertices and $2|\{X_{ij}|X_{ij} > 0\}|$ arcs. This implies that solving the separation problem for subtour elimination constraints requires, at most, $0(|V_0|^4)$ operations.

However, in practice, the size of the network that we need to consider can be reduced by applying the shrinking heuristic of Padberg and Grötschel [42]. Our computational experience has been that applying the shrinking heuristic results in the network having substantially less than $(|V_0| + 2)$ vertices. Note here that in the computational results reported below we used the maximum flow code due to Goldfarb and Grigoriadis [27].

4. PROBLEM REDUCTION

There are a number of reduction tests that have been given in the literature [2–4, 15–18] that can be used to reduce the size of the problem that we need to consider. In this section, we outline those that we used in the algorithm presented in this paper.

(1) Degree

As $|P_i|$ represents the degree of vertex i, we have

(a) any vertex $i \in V - K$ for which $|P_i| \leq 1$ can be deleted from the problem; and

(b) a vertex $i \in K$ with $|P_i| = 1$ implies that the single edge in P_i is in the Steiner tree if $|K| \geq 2$.

(2) Nearest Vertex

For any vertex $k \in K$, let $i \in V$ be the nearest vertex to k which is connected by an edge to k, and $j \in V$, be the second nearest such vertex. Letting d_{pq} represent the least cost of an elementary path from p to q, we have that the edge joining k to i is in the Steiner tree if

(a) $i \in K$; or

(b) $i \in V - K$ but there exists a vertex $p \in K$ ($p \neq k$) connected by an edge to i and $c_{ki} + c_{ip} \leq c_{kj}$; or

(c) $i \in V - K$ and $c_{ki} + \min\{d_{ip} | p \in K, p \neq k\} \leq c_{kj}$.

Note here that (a) and (b) are special cases of (c).

(3) Edge Deletion

For any edge $(i, j) \in E_0$ if there exists $k \in K$ such that $c_{ij} \geq \max(c_{ik}, c_{jk})$, then edge (i, j) can be deleted from the problem.

(4) Nearest Special Vertices (NSV) Test

Suppose that edge (i, j) is in an MST of the graph $G = (V, E)$. If edge (i, j) is deleted from G, then the MST of the graph remaining can be found by adding the minimum cost edge (v, w) between the two vertex subsets left after the removal of edge (i, j) from the MST. Then, edge (i, j) is in the Steiner tree if $c_{vw} \geq \min\{d_{ik} | k \in K, d_{ik} < d_{jk}\} + c_{ij} + \min\{d_{jk} | k \in K, d_{jk} < d_{ik}\}$. This test is due to Duin [15, 17].

(5) Special Distance (SD) Test

Define the concept of a special i–j path (for any $i, j \in V$) as any elementary path between i and j in the complete graph based on the vertex set $K^* = K \cup \{i\} \cup \{j\}$, where the cost of any edge (p, q) ($p \in K^*$, $q \in K^*$) in this graph is given by d_{pq}. Let T_{ij} be the set of all such paths and let Q_{ijt} $t \in T_{ij}$ be the set of edges in path t. The special distance s_{ij} is given by the minimum bottleneck edge cost over paths T_{ij}, i.e., $s_{ij} = \min\{\max\{d_{pq} | (p, q) \in Q_{ijt}\} t \in T_{ij}\}$. Then, the edge (i, j) can be deleted from the problem if $s_{ij} < c_{ij}$. This test is due to Duin [15, 17, 18].

(6) Local Special Distance (LSD) Test

Let $N_i = \{j | (i, j) \in P_i\}$ be the set of vertices adjacent to i. Then, any vertex $i \in V - K$ can be deleted from the problem if for every subset $N^* \subseteq N_i$ with $|N^*| \geq 3$ the cost of the MST of N^* using special distances (s_{ij}) is $\leq \sum_{j \in N^*} c_{ij}$. If vertex i is deleted, then an edge (p, q) of cost $c_{pi} + c_{iq}$ needs to be added for every $p \in N_i$, $q \in N_i - p$ with $s_{pq} = c_{pi} + c_{iq}$. This test is due to Duin [15, 17] and was implemented by him in [15] for all vertices $i \in V - K$ with $|N_i| < 8$. In the computational results presented later, we implemented it for all vertices $i \in V - K$ with $|N_i| \leq 5$.

(7) LP-based Tests

At any stage, either at the initial LP or after adding violated constraints to the initial LP and resolving, we have an LP solution $\{X_{ij}\}$ of value Z_{LB} (which is a lower bound on the optimal solution to the original SPG). Let Z_{UB} be any upper bound upon the optimal solution to the problem (e.g., derived from a heuristic). Then, the LP-based reduction tests that we used were the following:

(a) Reduced Cost

As we have an LP solution $\{X_{ij}\}$, each nonbasic variable in that solution has a reduced cost. Hence, any nonbasic variable x_{pq} (where currently $X_{pq} = 0$) for which (Z_{LB} + the reduced cost for x_{pq}) exceeds Z_{UB} can be eliminated from the problem (since if $x_{pq} = 1$, it would increase the lower bound above the upper bound Z_{UB}).

(b) Dual Variables

Limited computational experience indicated that relatively few variables (edges) were eliminated by the reduced cost test given above. Accordingly, we also used a reduction test based upon the dual variables.

As we have an LP solution, we have, for each constraint in the LP, a dual variable. Consider the complete strengthened formulation of the problem given previously [Eqs. (9)–(16)].

Suppose now that we relax, in a Lagrangean fashion (e.g., see [6, 23, 24]), constraints (10), (11), and (15) where we use as Lagrange multipliers for these constraints the corresponding dual variable values obtained from the

LP solution. Note here that, obviously, only exclusion constraints (15) that have been added to the LP are included in the Lagrangean relaxation.

Letting C_{ij} represent the cost, in the Lagrangean relaxation, of edge (i, j), the Lagrangean program is

$$\text{minimize} \quad \sum_{(i,j)\in E_0} C_{ij}x_{ij} + \text{a constant} \tag{22}$$

$$\text{subject to} \quad \sum_{p\in W\ q\in V_0-W\ (p,q)\in E_0} x_{pq} \geq 1 \quad 1 \in W$$
$$k \in V_0 - W \quad \forall W \subseteq V_0 \quad \forall k \in K - \{1\} \tag{23}$$

$$\sum_{(p,q)\in E_0} x_{pq} = |V_0| - 1 \tag{24}$$

$$\sum_{p,q\in T\ (p,q)\in E_0} x_{pq} \leq |T| - 1 \quad \forall T \subseteq V_0 \tag{25}$$

$$x_{ij} \in (0, 1) \quad \forall (i, j) \in E_0 \tag{26}$$

(where the precise nature of C_{ij} and the appropriate constant term in the objective function [Eq. (22)] need not concern us here).

The solution to this Lagrangean relaxation will yield a lower bound upon the optimal solution to the original problem. But constraints (23)–(26) merely specify that the set of edges chosen constitute a spanning tree on (V_0, E_0), i.e., we have a lower bound, derived from the LP solution that is simply to find the SST of the graph with edge costs $\{C_{ij}\}$ and, in fact, is essentially the same as the lower bound given in [4].

The difference between the lower bound presented in [4] and the lower bound given above is that

(a) in [4], we did not consider the degree constraints [constraint (10)];
(b) in [4], *all* the exclusion constraints were considered, whereas in the lower bound given above, we consider only a *subset* of the exclusion constraints, namely, those generated as violated constraints; and
(c) in [4], connectivity constraints were not used.

It is possible to use this lower bound [Eqs. (22)–(26)] in exactly the same manner as given in [4] to eliminate edges (variables) from the problem.

Informally, since the lower bound is an unconstrained SST problem, it is easily solved (e.g., use [34] or [48]). It is then simple to calculate the change in the SST for the addition or removal of an edge (i, j).

Suppose that we calculate the lower bound for the situation where an edge (i, j) is in the solution ($x_{ij} = 1$). If this lower bound exceeds Z_{UB}, then it is clear that edge (i, j) cannot appear in the optimal solution and so can be deleted from the problem.

A similar analysis applies in the case of forcing an edge out of solution ($x_{ij} = 0$). The mathematical details of how to carry out this test are given in [4] and so are not repeated here.

We would note here that the approach given above, namely, to use LP dual variable values as Lagrange multipliers in a partial Lagrangean relaxation (i.e., relax only a subset of the constraints) in order to derive variable penalty values for use in problem reduction, is an approach that could be used in any branch and cut algorithm. More about this approach can be found in [38]. Note also that similar ideas have been used by Fischetti and Vigo [22] in their branch and cut algorithm for the resource-constrained minimum-weight arborescence problem and by Fischetti and Toth [21] in their algorithm for the asymmetric traveling salesman problem.

(8) Compression

Note here that if, as a consequence of these reduction tests, we set x_{0j} to 1 (for some $j \in V - K$) then this implies that vertex j cannot be in the optimal Steiner tree, so vertex j (and the edges in P_j) can be deleted from the problem. Note also that if we set x_{ij} to 1 (for some $i \neq 0$ $i \in V, j \in V$), then we can ''compress'' the underlying graph (condense vertices i and j together and combine their associated edges P_i and P_j in an obvious fashion).

5. TREE SEARCH PROCEDURE

To find the optimal solution to SPG, we used a binary, depth-first, branch and cut tree search procedure, computing a lower bound from the LP (and adding violated constraints) at each tree node. The details of the procedure are given below:

5.1. Initial Tree Node

(1) Initial Reduction

We first reduced the problem using all the reduction tests given above (except for the LP-based tests) until no further reduction could be achieved.

(2) Root Vertex

We then chose the vertex to act as the root vertex (denoted by $1 \in K$ in this paper). Here, we used the same rule as in [4], namely, to choose the vertex $i \in K$ with the largest degree (ties broken arbitrarily).

(3) Upper Bound

Find a feasible solution (upper bound) by applying the algorithm of Takahashi and Matsuyama [50] as modified by Rayward-Smith and Clare [49] to the graph (V, E) with edge costs $\{c_{ij}\}$. If it is the first feasible solution found, or if it is an improved feasible solution (i.e., of cost less than Z_{UB}), then update Z_{UB} appropriately. Note here that the quality of the upper bound found by this heuristic is not crucial as we shall repeatedly (below) use LP solutions in order to seek improved feasible solutions.

(4) LP Solution

Solve the current LP (where initially the LP has no exclusion, connectivity, or subtour elimination constraints). As before, let the LP solution be given by $\{X_{ij}\}$ of value Z_{LB}. Note here that in the computational results reported below we used the CPLEX LP code [12].

(5) Constraint Removal

Examine the current LP solution and if there are any exclusion, connectivity, or subtour elimination constraints which are no longer active (i.e., have a nonzero slack variable), then remove these constraints from the problem. Note here that to avoid the possibility of cycling this removal was only done at iterations at which the LP solution value changed (increased).

(6) Feasible Solution

In an attempt to find an improved feasible solution, apply the algorithm of Takahashi and Matsuyama [50] as modified by Rayward-Smith and Clare [49] to the graph (V, E) with the cost for edge (i, j) given by $(1 - X_{ij})c_{ij}$. Conceptually, this implies that any edge (i, j) not in the current LP solution ($X_{ij} = 0$) retains its original cost and any edge (i, j) in the current LP solution ($X_{ij} > 0$) gets a cost lower than its original cost.

Let $R(\subseteq V)$ be the set of vertices used in the spanning tree given by applying the modified Takahashi and Matsuyama algorithm to this graph (where we will have $K \subseteq R$). Find the SST of R and prune it of any vertices $i \in R - K$. The tree left after this pruning will be a feasible solution to the original SPG. If it is an improved feasible solution (i.e., of cost less than Z_{UB}), then update Z_{UB} appropriately.

Note here that, conceptually, this means that we repeatedly search (over the course of the algorithm) for an improved feasible solution based upon the structure of the current LP solution. We believe that this approach, namely, using the structure of the LP solution in a systematic fashion to seek improved feasible solutions, is an approach that will be of value in other branch and cut algorithms.

(7) Reduction

Apply the degree, nearest vertex [(a) and (b)], edge deletion, and LP-based reduction tests given above in an attempt to reduce the size of the problem. Limited computational experience indicated that applying these tests at every iteration was too time-consuming and so they were only applied every fifth iteration.

(8) Restarting

Limited computational experience indicated that a good strategy was to restart the algorithm after a certain amount of reduction had been achieved. For the purposes of this paper, once the number of edges eliminated from the problem (by the reduction tests) exceeded $0.1|E_0|$ (i.e., 10% of the edges had been eliminated), then the problem was restarted [go to step (1) above].

The logic here is that by restarting the problem we can compress the graph that we need to consider (as mentioned under reduction tests above). This has a favorable impact: upon the size of the LP that we need to solve, upon the size of the directed network that we need to consider when solving the separation problem for subtour elimination constraints, and upon the maximum LP value achievable (see below).

In addition, restarting the problem can help to avoid the problem of ''tailing-off'' (smaller and smaller objective function changes) encountered in branch and cut algorithms (e.g., see [9, 43]). Without restarting, we may have no choice but to commence the tree search once ''tailing-off'' occurs.

(9) Violated Constraints

Examine the current LP solution $\{X_{ij}\}$ and

(a) add all violated exclusion constraints to the LP;
(b) add all violated connectivity constraints to the LP; and
(c) for each vertex $i \in V$ involved in a violated subtour elimination constraint, add a single subtour elimination constraint (the most violated constraint for i) to the LP.
(d) If any violated constraints have been added at (a), (b), or (c) above, go to step (4).

Note here that when we restart the problem [step (8) above] we do not carry forward any of the exclusion, connectivity, or subtour elimination constraints that have been previously added to the problem.

(10) Termination

The above procedure will terminate with a solution of value Z_{LP} (associated with variables $\{X_{ij}\}$) for which there are *no* violated exclusion, connectivity, or subtour elimination constraints. As such, this solution is the solution of the LP relaxation of the *complete* strengthened formulation of SPG [Eqs. (9)–(16)]. This solution has either

(a) $\{X_{ij}\}$ integer, i.e., $\{X_{ij}\}$, of value Z_{LP}, is the optimal solution to SPG; or
(b) $\{X_{ij}\}$ fractional and $Z_{LP} = Z_{UB}$, i.e., Z_{UB} is the optimal solution to SPG; or
(c) $\{X_{ij}\}$ fractional and $Z_{LP} < Z_{UB}$.

In case (c) above, we resolve the problem using a binary depth-first tree search procedure as detailed below.

5.2. Other Tree Nodes

We branched in the tree search by setting edges into solution, performing reduction tests, and adding violated constraints as appropriate. The details are given below:

(1) Forward Branching

In forward branching, we chose the (edge) variable x_{ij} which had a fractional value in the current LP solution and for which $|X_{ij} - 0.5|$ was a minimum and branched by setting that variable equal to 1.

(2) Lower Bound

At each tree node, we performed the same procedure [steps (3)–(10) inclusive] as given above for the initial tree node except that we did not do any restarts in the tree [i.e., no step (8)]. Note here that this typically means that

(a) violated constraints are added at each tree node—i.e., we have a *branch and cut* algorithm; and
(b) problem reduction takes place based upon the variables (edges) set as a result of branching.

(3) Backtracking

We can backtrack in the tree when $Z_{LB} \geq Z_{UB}$.

6. COMPUTATIONAL RESULTS

6.1. Test Problems

The branch and cut algorithm presented in this paper (henceforth, denoted by BC-BL) was programmed in FORTRAN and run on a Silicon Graphics Indigo workstation for a number of test problems drawn from the literature. These test problems were problem sets C, D, and E as solved by Beasley [4]. (As in [9], problem set B from [4] was easily solved by BC-BL and the results are not worth reporting here.) These test problems are publically available from the OR-Library [5] (E-mail the message *steininfo* to *o.rlibrary@ic.ac.uk* or see the www address *http://mscmga.ms.ic.ac.uk/jeb/orlib/steininfo.html*).

6.2. Results

Tables I–III give the results of BC-BL on problem sets C, D, and E, respectively. In those tables, we give, for each problem,

(a) the size of the problem after initial reduction;
(b) the number of violated exclusion, connectivity, and subtour elimination constraints (cuts) added (over all restarts) at the initial tree node;
(c) the number of restarts;
(d) the gap between the final LP solution (Z_{LP}) at the initial tree node and the optimal solution [as measured by 100 (optimal value − Z_{LP})/(optimal value)];
(e) the size of the final (reduced) problem at the initial tree node; and
(f) the computation time in Silicon Graphics Indigo CPU seconds for the initial tree node.

For those problems which did not terminate at the initial tree node, we also give

(a) the additional number of violated exclusion, connectivity, and subtour elimination constraints (cuts) added (over all tree search nodes);
(b) the number of tree search nodes; and
(c) the additional computation time in Silicon Graphics Indigo CPU seconds for these nodes.

For all problems, we give

(a) the *maximum* size (over all iterations/tree nodes) of the network involved in detecting violated subtour elimination constraints (see above);
(b) the *maximum* size (over all iterations/tree nodes) of the LP relaxation (as measured by the number of constraints); and
(c) the optimal solution value.

6.3. Effect of Duin Reductions

Recall that we included in BC-BL a number of reduction tests (nearest special vertices [NSV], special distance [SD], and local special distance [LSD]) due to Duin. To quantify the effects of these Duin reductions on our results, Tables IV–VI give the results of BC-BL, but without these reductions, on problem sets C, D, and E, respectively. Comparing Tables I–III and Tables IV–VI, it is apparent that

TABLE I. Computational results for BC-BL on problem set C

				Initial Reduction			Initial Tree Node							Tree Search			Maximum Network			
Problem No.	$\|V\|$	$\|E\|$	$\|K\|$	$\|V\|$	$\|E\|$	$\|K\|$	No. Cuts Exclusion: Connectivity: Subtour	No. Restarts	Gap (%)	$\|V\|$	$\|E\|$	$\|K\|$	Time	No. Additional Cuts Exclusion: Connectivity: Subtour	No. Tree Nodes	Additional Time	Nodes	Arcs	Maximum No. Constraints in the LP Relaxation	Optimal Solution Value
C-1	500	625	5	103	184	5	89:45:42	2	—	—	—	—	13	—	—	—	20	56	176	85
2			10	75	134	8	175:112:76	1	—	—	—	—	16	—	—	—	34	102	275	144
3			83	23	36	13	29:21:14	0	—	—	—	—	2	—	—	—	22	34	80	754
4			125	7	10	5	5:4:3	0	—	—	—	—	1	—	—	—	7	10	20	1079
5			250	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	—	1579
6	500	1000	5	348	800	5	1024:441:431	7	—	—	—	—	346	—	—	—	40	150	569	55
7			10	346	784	9	909:508:359	3	—	—	—	—	230	—	—	—	42	132	1073	102
8			83	42	80	14	73:52:33	0	—	—	—	—	11	—	—	—	34	78	152	509
9			125	72	132	36	135:91:116	0	—	—	—	—	23	—	—	—	67	268	246	707
10			250	12	17	8	12:7:12	0	—	—	—	—	9	—	—	—	12	20	38	1093
11	500	2500	5	487	1904	5	3485:1233:1637	10	—	—	—	—	2058	—	—	—	46	202	1666	32
12			10	452	1595	9	2384:1215:972	7	—	—	—	—	868	—	—	—	51	180	1325	46
13			83	138	334	32	534:301:269	2	—	—	—	—	65	—	—	—	84	320	553	258
14			125	5	7	4	0:0:3	0	—	—	—	—	13	—	—	—	—	—	14	323
15			250	—	—	—	—	—	—	—	—	—	17	—	—	—	—	—	—	556
16	500	12,500	5	500	3594	5	2446:334:1200	3	—	—	—	—	3655	—	—	—	43	92	3681	11
17			10	498	3324	8	4685:1358:2066	3	—	—	—	—	8095	—	—	—	52	202	4683	18
18			83	444	2291	47	16,035:4338:32,175	9	0.8850	159	501	40	42,900	3:1:8	4	78	365	2022	3857	113
19			125	379	1754	41	324:262:66	1	—	—	—	—	126	—	—	—	95	194	954	146
20			250	—	—	—	—	—	—	—	—	—	116	—	—	—	—	—	—	267

TABLE II. Computational results for BC-BL on problem set D

				Initial Reduction			Initial Tree Node							Tree Search			Maximum Network			
Problem No.	$\|V\|$	$\|E\|$	$\|K\|$	$\|V\|$	$\|E\|$	$\|K\|$	No. Cuts Exclusion: Connectivity: Subtour	No. Restarts	Gap (%)	$\|V\|$	$\|E\|$	$\|K\|$	Time	No. Additional Cuts Exclusion: Connectivity: Subtour	No. Tree Nodes	Additional Time	Nodes	Arcs	Maximum No. Constraints in the LP Relaxation	Optimal Solution Value
D-1	1000	1250	5	226	424	5	432:227:177	3	—	—	—	—	106	—	—	—	31	94	376	106
2			10	247	454	10	530:225:184	2	—	—	—	—	77	—	—	—	61	210	580	220
3			167	—	—	—	—	—	—	—	—	—	5	—	—	—	—	—	—	1565
4			250	13	18	8	11:9:6	0	—	—	—	—	5	—	—	—	13	20	38	1935
5			500	8	12	6	6:2:1	0	—	—	—	—	8	—	—	—	8	28	19	3250
6	1000	2000	5	737	1694	5	2867:1401:1125	6	5.7836	107	298	5	2088	254:70:157	12	103	85	532	1432	67
7			10	711	1634	10	1618:568:667	3	—	—	—	—	744	—	—	—	47	136	1718	103
8			167	103	178	47	186:118:151	0	—	—	—	—	90	—	—	—	97	352	333	1072
9			250	7	10	5	5:3:1	0	—	—	—	—	71	—	—	—	7	18	18	1448
10			500	15	28	11	11:5:9	0	—	—	—	—	52	—	—	—	15	26	37	2110
11	1000	5000	5	975	4190	5	6303:1198:2847	11	—	—	—	—	8076	—	—	—	84	484	2764	29
12			10	950	3568	9	4353:1382:1665	7	2.3810	11	26	5	3446	1:1:0	2	1	66	346	2265	42
13			167	12	20	7	14:9:8	0	—	—	—	—	111	—	—	—	11	20	40	500
14			250	—	—	—	—	—	—	—	—	—	110	—	—	—	—	—	—	667
15			500	8	13	6	4:2:0	0	—	—	—	—	114	—	—	—	8	10	17	1116
16	1000	25,000	5	1000	8296	5	466:270:198	3	—	—	—	—	1200	—	—	—	24	70	1276	13
17			10	999	7997	9	5111:392:2412	1	—	—	—	—	17,856	—	—	—	71	158	8181	23
18			167	896	4737	94	12,115:4124:9654	2	—	—	—	—	14,968	—	—	—	768	4328	7806	223
19			250	801	4112	97	3802:1289:2344	0	—	—	—	—	20,893	—	—	—	342	1666	6669	310
20			500	—	—	—	—	—	—	—	—	—	791	—	—	—	—	—	—	537

TABLE III. Computational results for BC-BL on problem set E

				Initial Reduction			Initial Tree Node							Tree Search			Maximum Network			
Problem No.	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert K\rvert$	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert K\rvert$	No. Cuts Exclusion: Connectivity: Subtour	No. Restarts	Gap (%)	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert K\rvert$	Time	No. Additional Cuts Exclusion: Connectivity: Subtour	No. Tree Nodes	Additional Time	Nodes	Arcs	Maximum No. Constraints in the LP Relaxation	Optimal Solution Value
E-1	2500	3125	5	647	1234	5	1357:295:650	5	—	—	—	—	1028	—	—	—	30	68	1716	111
2			10	666	1244	9	3079:1525:1295	4	—	—	—	—	1765	—	—	—	64	310	2530	214
3			417	81	135	46	115:77:120	0	—	—	—	—	113	—	—	—	79	222	248	4013
4			625	19	33	13	17:9:6	0	—	—	—	—	75	—	—	—	19	34	48	5101
5			1250	7	10	5	5:4:3	0	—	—	—	—	99	—	—	—	7	10	19	8128
6	2500	5000	5	1816	4268	5	8846:2028:4179	8	—	—	—	—	8547	—	—	—	51	168	5792	73
7			10	1861	4322	10	17,461:4000:1107	8	0.3448	341	652	10	11,801	1:1:0	4	52	492	1630	8430	145
8			417	188	349	90	371:234:470	0	—	—	—	—	1204	—	—	—	171	464	642	2640
9			625	78	136	42	94:61:35	0	—	—	—	—	1083	—	—	—	72	152	193	3604
10			1250	11	18	8	10:4:4	0	—	—	—	—	814	—	—	—	11	20	28	5600
11	2500	12,500	5	2479	11,479	5	2350:1337:757	8	—	—	—	—	10,426	—	—	—	56	268	3005	34
12			10	2455	10,834	10							(21,600)							
13			417	300	655	116	2041:890:3517	2	—	—	—	—	7484	—	—	—	280	1128	1128	1280
14			625	34	61	19	45:33:22	0	—	—	—	—	1806	—	—	—	33	68	113	1732
15			1250	—	—	—	—	—	—	—	—	—	1599	—	—	—	—	—	—	2784
16	2500	62,500	5	2500	23,397	5	1192:639:448	4	—	—	—	—	13,517	—	—	—	34	98	3524	15
17			10	2500	21,783	10							(21,600)							
18			417	2259	12,255	247							(21,600)							
19			625	1777	8173	183							(21,600)							
20			1250	—	—	—	—	—	—	—	—	—	10,277	—	—	—	—	—	—	1342

A time in parentheses signifies that the algorithm did not finish in the time shown.

TABLE IV. Computational results for BC-BL (without Duin reductions) on problem set C

				Initial Reduction			Initial Tree Node							Tree Search			Maximum Network		
Problem No.	$\|V\|$	$\|E\|$	$\|K\|$	$\|V\|$	$\|E\|$	$\|K\|$	No. Cuts Exclusion: Connectivity: Subtour	No. Restarts	Gap (%)	$\|V\|$	$\|E\|$	$\|K\|$	Time	No. Additional Cuts Exclusion: Connectivity: Subtour	No. Tree Nodes	Additional Time	Nodes	Arcs	Maximum No. Constraints in the LP Relaxation
C-1	500	625	5	138	241	5	172:100:91	3	—	—	—	—	16	—	—	—	23	56	220
2			10	128	228	8	465:289:219	5	—	—	—	—	34	—	—	—	34	108	423
3			83	103	162	31	276:155:125	3	—	—	—	—	12	—	—	—	69	150	319
4			125	87	132	30	204:135:96	1	—	—	—	—	10	—	—	—	63	128	273
5			250	18	23	11	15:7:7	0	—	—	—	—	1	—	—	—	16	20	47
6	500	1000	5	366	830	5	1637:679:755	9	—	—	—	—	352	—	—	—	61	196	605
7			10	381	848	9	1326:825:515	4	—	—	—	—	270	—	—	—	45	162	997
8			83	342	731	53	2148:1071:1125	6	—	—	—	—	273	—	—	—	146	338	1215
9			125	317	644	68	2476:1182:2144	3	0.2829	178	395	63	1052	209:140:678	66	1153	170	714	1080
10			250	51	77	22	53:32:20	0	—	—	—	—	7	—	—	—	39	62	141
11	500	2500	5	500	1967	5	2239:761:1176	10	—	—	—	—	1789	—	—	—	56	172	1742
12			10	497	1856	9	3377:1549:1381	10	—	—	—	—	1065	—	—	—	50	172	1307
13			83	425	1301	48	2517:1091:1409	3	0.7623	173	475	41	761	328:382:749	50	1841	146	754	1421
14			125	383	1068	62	966:500:544	1	—	—	—	—	131	—	—	—	152	386	1309
15			250	85	158	22	145:121:59	2	—	—	—	—	21	—	—	—	48	138	232
16	500	12,500	5	500	3661	5	144:108:83	2	—	—	—	—	174	—	—	—	17	40	592
17			10	498	3535	8	2390:417:1097	1	—	—	—	—	3039	—	—	—	57	116	4095
18			83	463	2718	47	21,486:5844:33,229	17	0.8850	59	101	30	30,344	0:0:0	4	31	371	2014	3699
19			125	416	1952	41	571:451:176	2	—	—	—	—	130	—	—	—	89	152	1000
20			250	190	521	10	63:56:16	0	—	—	—	—	142	—	—	—	23	38	314

TABLE V. Computational results for BC-BL (without Duin reductions) on problem set D

														Tree Search					
							Initial Tree Node												
				Initial Reduction													Maximum Network		
Problem No.	|V|	|E|	|K|	|V|	|E|	|K|	No. Cuts Exclusion: Connectivity: Subtour	No. Restarts	Gap (%)	|V|	|E|	|K|	Time	No. Additional Cuts Exclusion: Connectivity: Subtour	No. Tree Nodes	Additional Time	Nodes	Arcs	Maximum No. Constraints in the LP Relaxation
D-1	1000	1250	5	276	508	5	858:530:361	9	3.7736	14	26	5	142	13:12:5	8	2	33	112	499
2			10	287	518	10	647:274:224	2	—	—	—	—	81	—	—	—	69	282	687
3			167	177	273	54	363:241:150	2	—	—	—	—	21	—	—	—	116	208	539
4			250	99	144	41	140:72:70	0	—	—	—	—	12	—	—	—	78	200	298
5			500	48	62	26	58:42:163	0	—	—	—	—	29	—	—	—	45	142	193
6	1000	2000	5	759	1727	5	2994:1431:1181	7	5.2239	101	272	5	1558	140:59:139	8	54	80	380	1352
7			10	747	1700	10	1595:589:645	3	—	—	—	—	620	—	—	—	49	148	1811
8			167	707	1501	113	3636:2025:2089	4	—	—	—	—	717	—	—	—	319	710	2395
9			250	613	1294	136	4299:2501:3094	8	—	—	—	—	1219	—	—	—	326	816	2036
10			500	294	623	92	1021:651:903	4	—	—	—	—	190	—	—	—	177	370	830
11	1000	5000	5	992	4297	5	6295:1452:3035	12	—	—	—	—	5943	—	—	—	53	128	3016
12			10	995	4031	9	3245:954:1347	8	2.3810	8	12	4	2316	1:1:0	2	1	67	352	2816
13			167	891	2855	108	7620:4176:4892	8	—	—	—	—	2107	—	—	—	297	792	2959
14			250	759	2141	115	6341:3460:4293	6	—	—	—	—	1554	—	—	—	289	634	2640
15			500	210	425	56	731:410:315	3	—	—	—	—	149	—	—	—	117	222	690
16	1000	25,000	5	1000	8338	5	509:348:249	4	—	—	—	—	1128	—	—	—	24	82	1267
17			10	999	8308	9	5315:580:2852	2	—	—	—	—	16,928	—	—	—	745	3964	8168
18			167	927	5948	94	13,951:3598:10,582	2	—	—	—	—	15,223	—	—	—	787	4486	7406
19			250	841	4559	97	4477:1205:6013	0	—	—	—	—	9020	—	—	—	685	4054	6397
20			500	418	1142	33	218:168:49	0	—	—	—	—	905	—	—	—	80	154	828

TABLE VI. Computational results for BC-BL (without Duin reductions) on problem set E

				Initial Reduction			Initial Tree Node							Tree Search			Maximum Network		
Problem No.	$\|V\|$	$\|E\|$	$\|K\|$	$\|V\|$	$\|E\|$	$\|K\|$	No. Cuts Exclusion: Connectivity: Subtour	No. Restarts	Gap (%)	$\|V\|$	$\|E\|$	$\|K\|$	Time	No. Additional Cuts Exclusion: Connectivity: Subtour	No. Tree Nodes	Additional Time	Nodes	Arcs	Maximum No. Constraints in the LP Relaxation
E-1	2500	3125	5	678	1280	5	1978:453:956	6	—	—	—	—	1131	—	—	—	35	90	1990
2			10	705	1305	9	3575:1618:1522	5	—	—	—	—	1555	—	—	—	67	316	2452
3			417	492	781	145	1900:1173:3853	2	—	—	—	—	2147	—	—	—	310	716	1565
4			625	160	225	64	385:244:150	2	—	—	—	—	61	—	—	—	119	224	463
5			1250	30	30	18	28:20:9	0	—	—	—	—	55	—	—	—	28	68	84
6	2500	5000	5	1844	4331	5	5490:1402:2594	6	—	—	—	—	13,050	—	—	—	47	162	5707
7			10	1887	4344	10	17,730:4930:7778	14	—	—	—	—	10,243	—	—	—	106	234	8133
8			417	1691	4022	265							(21,600)						
9			625	1514	3723	301							(21,600)						
10			1250	584	1387	188	2698:2174:4906	4	—	—	—	—	4483	—	—	—	357	814	1736
11	2500	12,500	5	2498	11,631	5	4863:2190:1436	11	—	—	—	—	17,103	—	—	—	154	902	3529
12			10	2498	11,500	10							(21,600)						
13			417	2274	8249	282							(21,600)						
14			625	2004	6973	302							(21,600)						
15			1250	802	2073	166	3159:1981:8570	5	—	—	—	—	10,294	—	—	—	327	904	3746
16	2500	62,500	5	2500	23,553	5	684:485:322	4	—	—	—	—	13,717	—	—	—	28	76	2878
17			10	2500	23,064	10							(21,600)						
18			417	2321	17,634	247							(21,600)						
19			625	2042	9663	190							(21,600)						
20			1250	1088	3084	84							(21,600)						

A time in parentheses signifies that the algorithm did not finish in the time shown.

(a) the Duin reductions lead to smaller graphs which make the branch and cut approach computationally more feasible;

(b) the Duin reductions do not always lead to smaller computation times—of the 51 problems solved to optimality in Tables IV–VI, 16 problems took more computation time with the Duin reductions; and

(c) the gap between the final LP solution (Z_{LP}) at the initial tree node and the optimal solution is relatively unaffected by the Duin reductions. Of the 51 problems solved to optimality in Tables IV–VI, 45 problems have a zero gap and the average gap for the remaining six problems is 2.2181%. The corresponding figures for Tables I–III are that of the 56 problems solved to optimality 52 problems have zero gap and the average gap for the remaining four problems is 2.3486%.

We would conclude here that the overall effect of the Duin reductions is on computation time, with relatively little effect on the gap between the final LP solution value and the integer optimal solution. Note also here that the Duin reductions that we implemented in BC-BL were a subset of the full set of reductions presented by Duin in [15] and that we may not have implemented them in the computationally most effective manner.

6.4. Upper and Lower Bounds

Examining Tables I–VI, it is clear that a large number of problems are solved without any branching being necessary (irrespective of whether the Duin reductions are applied or not). This occurs because

(a) the heuristics that we apply are very successful at finding the optimal integer solution; and

(b) the lower bound (as given by the LP solution value) frequently coincides with the optimal integer solution value.

We believe that the quality of the upper bounds that we achieve are due to the approach that we adopted, namely, using the structure of the LP solution in a systematic fashion to seek improved feasible solutions. As an illustration of this, we have that for 55 of the 56 problems solved in Tables I–III the upper bound is optimal at the initial tree node. In other words, for only one problem did we fail to discover the optimal solution at the initial tree node.

6.5. Shrinking Heuristic

The success of the shrinking heuristic of Padberg and Grötschel [42], in terms of the size of the network flow problem solved to identify violated subtour elimination constraints, can be gauged from the figures for the maximum network size in Tables I–VI. More specifically, the effect of the shrinking heuristic over the 98 problems in Tables I–VI for which maximum network size figures are available has been to reduce network size (in terms of number of vertices) by 57% on average. For 60 larger problems (original reduced graph $\geq$ 250 vertices), this reduction increases to an average of 74%.

6.6. Restarting

To illustrate the advantage of the restarting strategy, we plot for problem C-11 in Figure 2 (taken from Table IV, i.e. without Duin reductions) the value of the LP relaxation at each iteration (at the initial tree node) as given by the original algorithm BC-BL and as given by that algorithm with no restarts. It can be seen from that figure that BC-BL reaches the optimal solution of 32 after 10 restarts (and after approximately 650 iterations). However, with no restarts, the maximum lower bound achieved is only 30 (after approximately 450 iterations), necessitating branching. *In other words*, without restarts, we are left with a sizable gap ($\frac{2}{32}$ = 6.25%) at the initial tree node and require branching.

To clarify what has happened here—the combined effect of problem reduction (fixing variables) over successive restarts has been to *increase* (from 30 to 32) the maximum solution value obtainable from the LP relaxation [Eqs. (9)–(16)] of the complete strengthened formulation of SPG.

We would emphasize here that this illustrates that our strategy of using LP dual variable values in a partial Lagrangean relaxation for problem reduction can yield significant benefits. In particular, note from Tables I–VI the size of the final (reduced) problem at the initial tree node for those (few) problems that required branching.

We also show in Figure 2 BC-BL with no restarts and

(a) no degree 2 constraints [Eq. (10)], a maximum lower bound of 30 (after approximately 200 iterations);

(b) no connectivity constraints [Eq. (12)], a maximum lower bound of 30 (after approximately 400 iterations); and

(c) no subtour elimination constraints [Eq. (14)], a maximum lower bound of 29.764 (after approximately 300 iterations).

It appears from Figure 2 that there are a number of horizontal portions to the lines plotted. These are not plotting artifacts but are genuine; in other words, we often encounter situations in which the LP value remains unchanged for a considerable number of iterations.

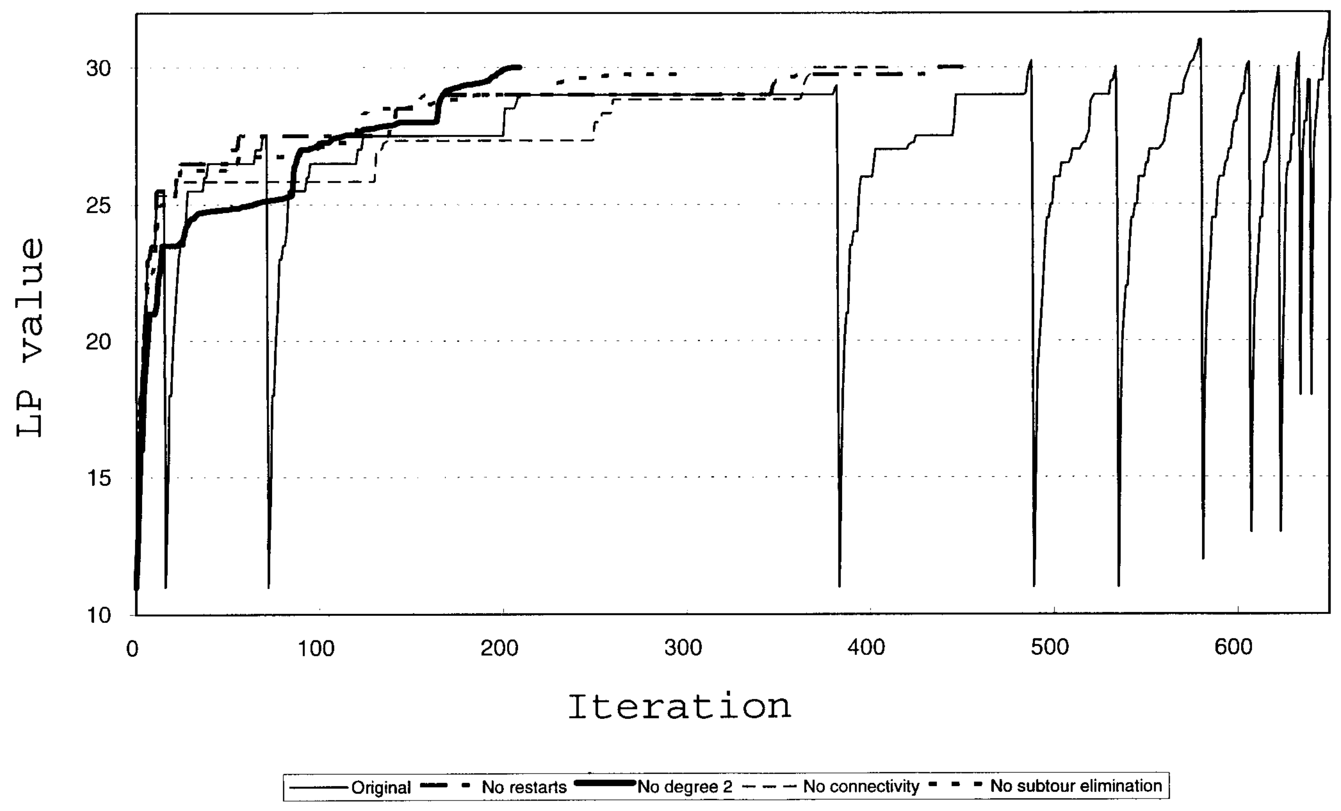

Fig. 2. Problem C-11.

6.7. Comparison

In Table VII, we compare BC-BL (with Duin reductions, Tables I–III) with the results of other workers. In that table, we present the gap at the initial tree node (if known) and the total computation time in seconds. However, all the computers used were different, which makes a *precise* comparison of these results difficult. Chopra et al. [9] in their branch and cut algorithm (henceforth, denoted by BC-CGR) used a Vax 8700; Beasley [4], a Cray X-MP/48; Lucena [37], a Sun Sparcstation 10; and Duin [15], a 486 66 MHz pc.

From Dongarra [13], we can make an *approximate* comparison of the results shown in Table VII by using scaling values for the achievable speeds of these computers as 15 for the Silicon Graphics Indigo, 0.99 for the Vax 8700, 121 for the Cray X-MP/48, 7 for the Sun Sparcstation 10, and 0.56 for the 486 66 MHz pc. Using these values, we would make the following observations:

(a) Among BC-BL, BC-CGR, and Lucena, no clear computational winner emerges: BC-BL is faster than BC-CGR in solving 26 problems, BC-BL is faster than Lucena in solving 23 problems, and Lucena is faster than BC-CGR in solving 28 problems;

(b) BC-BL, BC-CGR, and Lucena are all generally superior, not only in computation time terms but also in gap terms, to the Lagrangean relaxation algorithm of Beasley [4]; and

(c) although the performance of Duin [15] on problem set *E* is unknown, it is clear that its computation performance on problems sets C and D is superior to all other algorithms.

6.8. Duin [15] Results

The performance of Duin [15] deserves further comment. We believe that the key to his success is twofold:

1. His reduction tests are very effective.
2. His computational implementation is outstandingly effective.

In support of this, we would note that BC-BL includes a subset of the full set of Duin reductions [15], yet it is clear from the comparable problems solved by BC-BL purely by reduction (problems C-5/15/20, D-3/14/20) that our computation times are far inferior to those of Duin. Indeed, for these particular problems, our total computation time is 1040 Silicon Graphics Indigo seconds while the computation time for Duin for these problems

TABLE VII. Comparison of algorithms

	This Paper BC-BL		Chopra et al. [9] BC-CGR		Beasley [4]		Lucena [37]		Duin [15]	
Problem No.	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)
C-1	—	13	—	27.3	10.9837	113.57	—	8.2	—	0.7
2	—	16	—	811.7	2.5526	5.84	—	28.4	—	0.4
3	—	2	—	543.4	0.5906	152.78	—	9.9	—	0.4
4	—	1	—	509.6	—	3.61	—	7.5	—	0.4
5	—	1	—	473.9	—	2.73	—	3.0	—	0.7
6	—	346	—	48.9	10.4664	48.55	—	131.0	—	1.9
7	—	230	—	83.2	—	4.44	—	65.1	—	2.1
8	—	11	—	674.4	—	8.63	—	85.5	—	0.7
9	—	23	—	1866.3	0.2966	198.97	—	122.2	—	0.8
10	—	9	—	245.6	—	4.53	—	18.4	—	0.8
11	—	2058	—	333.3	12.1398	188.02	—	160.0	—	4.2
12	—	868	—	119.8	4.0712	25.04	—	129.0	—	3.3
13	—	65	Not known	9170.3	0.6968	166.53	—	156.7	1.6	1.5
14	—	13	—	211.7	—	8.67	—	117.5	—	0.9
15	—	17	—	210.6	—	7.30	—	56.6	—	1.0
16	—	3655	—	10.1	16.4790	32.37	—	154.1	—	7.2
17	—	8095	—	98.0	5.7762	24.17	—	127.1	—	6.4
18	0.8850	43,265	Not known	45,847.7	1.0473	104.34	—	524.2	9.6	247.7
19	—	126	—	116.9	—	86.48	—	114.1	5.2	34.8
20	—	116	—	14.9	—	157.80	—	125.0	—	2.9
D-1	—	106	—	475.6	9.0015	226.27	—	81.9	—	1.8
2	—	77	—	283.5	3.5551	252.47	—	36.6	—	1.7
3	—	5	—	2290.1	0.0130	21.85	—	36.0	—	0.9
4	—	5	—	3529.0	—	11.71	—	16.9	—	1.1
5	—	8	—	810.6	0.0076	11.76	—	18.4	—	2.4
6	5.7836	2191	—	2339.5	12.5762	4056.69	12.6866	678.9	—	7.4
7	—	744	—	99.7	1.9575	18.71	—	248.6	—	7.1
8	—	90	—	6984.5	0.0417	475.14	—	389.7	—	2.2
9	—	71	—	4629.7	0.0162	243.48	—	282.3	—	1.8
10	—	52	—	1312.1	—	20.21	—	167.5	—	2.6
11	—	8076	—	1374.4	18.0027	3290.48	—	806.8	—	13.5
12	2.3810	3447	—	305.0	4.0179	48.04	—	888.5	—	10.5
13	—	111	—	1864.0	0.0118	36.06	—	487.6	—	2.7
14	—	110	—	3538.4	0.1376	443.36	—	598.1	—	2.3
15	—	114	—	1409.7	0.0170	32.25	—	406.0	—	3.2
16	—	1200	—	871.3	12.0950	161.43	—	437.6	—	17.8
17	—	17,856	—	6965.2	8.5057	277.20	—	604.8	—	20.7
18	—	14,968	Not known	245,192.1	0.2538	222.15	—	730.5	4.8	1988.4
19	—	20,893	—	878.3	—	256.15	—	647.4	4.5	1420.0
20	—	791	—	47.1	—	1023.60	—	498.9	—	7.4
E-1	—	1028	—	1149.6	9.8495	1116.80	—	203.2		
2	—	1765	—	6251.2	4.8827	7124.10	—	437.8		
3	—	113	—	26,468.4	0.0218	1364.05	—	202.2		

Table VII continues

TABLE VII. Continued

	This Paper BC-BL		Chopra et al. [9] BC-CGR		Beasley [4]		Lucena [37]		Duin [15]	
Problem No.	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)
4	—	75	—	46,007.6	0.0339	378.66	—	206.7		
5	—	99	—	12,564.1	—	98.22	—	215.1		
6	—	8547	—	678.0	10.5719	1760.49	—	2860.0		
7	0.3448	11,853	—	27,124.0	14.0949	(21,600)	—	2852.2		
8	—	1204	—	118,617.5	0.0246	4459.30	—	3198.4		
9	—	1083	—	24,527.8	0.0434	18,818.53	—	2235.5		
10	—	814	—	39,260.7	0.0161	311.87	—	1797.1		
11	—	10,426	—	1900.6	9.9758	3061.45	—	5319.0		
12		(21,600)	—	7199.7	9.4853	(21,600)	—	6159.8		
13	—	7484	—	207,058.6	0.9272	(21,600)	—	9311.9		
14	—	1806	—	29,262.6	0.2527	(21,600)	—	6729.6		
15	—	1599	—	7666.0	0.0161	457.98	—	4611.8		
16	—	13,517	—	179.0	16.0455	7880.44	—	3570.4		
17		(21,600)	—	36,039.9	4.6859	445.69	—	4729.7		
18		(21,600)	Not known	Not known	16.3308	(21,600)	0.8979	(50,000)		
19		(21,600)	—	6371.8	0.5759	(21,600)	—	18,345.0		
20	—	10,277	—	272.2	—	14,037.13	—	15,832.5		

is 15.2 486 66 MHz pc seconds. Given the Dongarra [13] scaling factors of 15 and 0.56, respectively, these times equate to a ratio of 1040(15):15.2(0.56) = 1833:1, i.e., our best estimate is that our implementation is *at least* three orders of magnitude slower than the implementation of Duin.

We would also comment that our experimental evidence (presented above) has been that the effect of the subset of the full set of Duin reductions [15] incorporated into BC-BL is principally upon the computation time and not on the gap between the final LP solution value and the integer optimal solution. In particular, note that the gaps found by BC-BL are much smaller than are the gaps reported by Duin using a lower bound due to Wong [57], which from Table VII average to a gap figure of 0.6425% over the 40 problems in problem sets C and D. This compares with an average for BC-BL of 0.2262% and an average for Lucena of 0.3172%.

7. CONCLUSIONS

In this paper, we have presented a branch and cut algorithm for the Steiner problem in graphs. Computational results indicated that the algorithm developed is able to solve a large number of problems without resorting to branching. In addition, a number of issues, specifically,

- using LP dual variable values in a partial Lagrangean relaxation,
- using the structure of the LP solution to look for feasible solutions, and
- restarting

relating to branch and cut algorithms in general were highlighted.

REFERENCES

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

[2] A. Balakrishnan and N. R. Patel, Problem reduction methods and a tree generation algorithm for the Steiner network problem. *Networks* **17** (1987) 65–85.

[3] J. E. Beasley, An algorithm for the Steiner problem in graphs. *Networks* **14** (1984) 147–159.

[4] J. E. Beasley, An SST-based algorithm for the Steiner problem in graphs. *Networks* **19** (1989) 1–16.

[5] J. E. Beasley, OR-Library: Distributing test problems by electronic mail. *J. Oper. Res. Soc.* **41** (1990) 1069–1072.

[6] J. E. Beasley, Lagrangean relaxation. *Modern Heuristic*

Techniques for Combinatorial Problems (C. R. Reeves, Ed.). Blackwell, Oxford (1993) 243–303.

[7] P. Berman and V. Ramaiyer, Improved approximations for the Steiner tree problem. *J. Alg.* **17** (1994) 381–408.

[8] S. Chopra and E. R. Gorres, On the node weighted Steiner tree problem. Working paper. Available from the first author at Department of Managerial Economics and Decision Sciences, J. L. Kellogg Graduate School of Management, Northwestern University, Evanston IL 60208 (1990).

[9] S. Chopra, E. R. Gorres, and M. R. Rao, Solving the Steiner tree problem on a graph using branch and cut. *ORSA J. Comput.* **4** (1992) 320–335.

[10] S. Chopra and M. R. Rao, The Steiner tree problem I: Formulations, compositions and extension of facets. *Math. Prog.* **64** (1994) 209–229.

[11] S. Chopra and M. R. Rao, The Steiner tree problem II: Properties and classes of facets. *Math. Prog.* **64** (1994) 231–246.

[12] CPLEX Optimization Inc., Using the CPLEX Callable Library. CPLEX Optimization Inc., Suite 279, 930 Tahoe Blvd., Bldg. 802, Incline Valley, NV 89451-9436 (1995).

[13] J. J. Dongarra, Performance of various computers using standard linear equations software. Working paper. Available from the author at Computer Science Department, University of Tennessee, Knoxville, TN 37996-1301 (1995).

[14] K. A. Dowsland, Hill-climbing, simulated annealing and the Steiner problem in graphs. *Eng. Opt.* **17** (1991) 91–107.

[15] C. W. Duin, Steiner's problem in graphs: Approximation, reduction, variation. PhD Thesis. Institute of Actuarial Science & Economics, University of Amsterdam, Roetersstraat 18, 1018 WB Amsterdam, The Netherlands (1994).

[16] C. W. Duin and A. Volgenant, Some generalizations of the Steiner problem in graphs. *Networks* **17** (1987) 353–364.

[17] C. W. Duin and A. Volgenant, Reduction tests for the Steiner problem in graphs. *Networks* **19** (1989) 549–567.

[18] C. W. Duin and A. Volgenant, An edge elimination test for the Steiner problem in graphs. *Oper. Res. Lett.* **8** (1989) 79–83.

[19] C. Duin and S. Voß, Efficient path and vertex exchange in Steiner tree algorithms. *Networks* **29** (1997) 89–105.

[20] H. Esbensen, Computing near-optimal solutions to the Steiner problem in a graph using a genetic algorithm. *Networks* **26** (1995) 173–185.

[21] M. Fischetti and P. Toth, A polyhedral approach to the asymmetric traveling salesman problem. *Manag. Sci.*, to appear.

[22] M. Fischetti and D. Vigo, A branch-and-cut algorithm for the resource-constrained minimum-weight arborescence problem. *Networks* **29** (1997) 55–67.

[23] M. L. Fisher, The Lagrangian relaxation method for solving integer programming problems. *Manag. Sci.* **27** (1981) 1–18.

[24] M. L. Fisher, An applications oriented guide to Lagrangian relaxation. *Interfaces* **15** (1985) 10–21.

[25] R. Floren, A note on ''A faster approximation algorithm for the Steiner problem in graphs.'' *Inform. Process. Lett.* **38** (1991) 177–178.

[26] M. X. Goemans and Y. Myung, A catalog of Steiner tree formulations. *Networks* **23** (1993) 19–28.

[27] D. Goldfarb and M. D. Grigoriadis, A computational comparison of the Dinic and network simplex methods for maximum flow. *Ann. Oper. Res.* **13** (1988) 83–123.

[28] F. K. Hwang and D. S. Richards, Steiner tree problems. *Networks* **22** (1992) 55–89.

[29] F. K. Hwang, D. S. Richards, and P. Winter, *The Steiner Tree Problem.* North-Holland, Amsterdam (1992).

[30] A. Kapsalis, V. J. Rayward-Smith, and G. D. Smith, Solving the graphical Steiner tree problem using genetic algorithms. *J. Oper. Res. Soc.* **44** (1993) 397–406.

[31] B. N. Khoury and P. M. Pardalos, A heuristic for the Steiner problem in graphs. Working paper. Available from the authors at Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL 32611 (1993).

[32] B. N. Khoury, P. M. Pardalos, and D.-Z. Du, A test problem generator for the Steiner problem in graphs. *ACM Trans. Math. Soft.* **19** (1993) 509–522.

[33] B. N. Khoury, P. M. Pardalos, and D. W. Hearn, Equivalent formulations for the Steiner problem in graphs. *Network Optimization Problems: Algorithms, Applications and Complexity* (D.-Z. Du and P. M. Pardalos, Eds.). World Scientific, NJ (1993) 111–123.

[34] J. B. Kruskal, On the shortest spanning subtree of a graph and the travelling salesman problem. *Proc. Am. Math. Soc.* **7** (1956) 48–50.

[35] A. Lucena, Steiner problem in graphs: Lagrangean relaxation and cutting-planes. *COAL Bull.* **21** (1992) 2–7.

[36] A. Lucena, Tight bounds for the Steiner problem in graphs. Working paper. Available from the author at Laboratorio Nacional de Computacao Cientifica/CNPq, Rua Lauro Muller 455, Rio de Janeiro RJ 22290-060, Brazil (1993).

[37] A. Lucena, Steiner problem in graphs: Lagrangean relaxation and strong valid inequalities. Working paper. Available from the author at Laboratorio Nacional de Computacao Cientifica/CNPq, Rua Lauro Muller 455, Rio de Janeiro RJ 22290-060, Brazil (1993).

[38] A. Lucena and J. E. Beasley, Branch and cut algorithms. *Advances in Linear and Integer Programming* (J. E. Beasley, Ed.). Oxford University Press, Oxford (1996) 187–221.

[39] T. Matsui and K. Yabe, Edge cover lower bounds for the Steiner problem in graphs. Working paper. Available from the first author at Department of Industrial Administration, Science University of Tokyo, Noda, Chiba 278, Japan (1990).

[40] K. Mehlhorn, A faster approximation algorithm for the Steiner problem in graphs. *Inform. Process. Lett.* **27** (1988) 125–128.

[41] G. L. Nemhauser and L. A. Wolsey, Integer and combinatorial optimization. Wiley, New York (1988).

[42] M. W. Padberg and M. Grötschel, Polyhedral computations. *The Travelling Salesman Problem* (E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, Eds.). Wiley, New York (1985) 307–360.

[43] M. W. Padberg and G. Rinaldi, A branch-and-cut algorithm for the resolution of large-scale symmetric travelling salesman problems. *SIAM Rev.* **33** (1991) 60–100.

[44] M. W. Padberg and L. A. Wolsey, Trees and cuts. *Ann. Discr. Math.* **17** (1983) 511–517.

[45] J. Plesnik, A bound for the Steiner tree problem in graphs. *Math. Slov.* **31** (1981) 155–163.

[46] J. Plesnik, Heuristics for the Steiner problem in graphs. *Discr. Appl. Math.* **37/38** (1992) 451–463.

[47] C. Pornavalai, N. Shiratori, and G. Chakraborty, Neural network for optimal Steiner tree computation. *Neural Process. Lett.* **3** (1996) 139–149.

[48] R. C. Prim, Shortest connection networks and some generalisations. *Bell Syst. Tech. J.* **36** (1957) 1389–1401.

[49] V. J. Rayward-Smith and A. Clare, On finding Steiner vertices. *Networks* **16** (1986) 283–294.

[50] H. Takahashi and A. Matsuyama, An approximate solution for the Steiner problem in graphs. *Math. Jpn.* **6** (1980) 573–577.

[51] M. G. A. Verhoeven, M. E. M. Severens, and E. H. L. Aarts, Local search for Steiner trees in graphs. *Modern Heuristic Search Methods* (V. J. Rayward-Smith, I. H. Osman, C. R. Reeves, and G. D. Smith, Eds.). Wiley, New York (1996) 117–129.

[52] S. Voß, Steiner's problem in graphs: Heuristic methods. *Discr. Appl. Math.* **40** (1992) 45–72.

[53] S. Voß, Problems with generalized Steiner problems. *Algorithmica* **7** (1992) 333–335.

[54] S. Voss, Worst-case performance of some heuristics for Steiner's problem in directed graphs. *Inform. Process. Lett.* **48** (1993) 99–105.

[55] A. S. C. Wade and V. J. Rayward-Smith, Effective local search techniques for the Steiner tree problem. Working paper. Available from the second author at the School of Information Systems, University of East Anglia, Norwich NR4 7TJ, England (1996).

[56] P. Winter and J. M. Smith, Path-distance heuristics for the Steiner problem in undirected networks. *Algorithmica* **7** (1992) 309–327.

[57] R. T. Wong, A dual ascent approach for Steiner tree problems on a directed graph. *Math. Prog.* **28** (1984) 271–287.