
PERGAMON Mathematical and Computer Modelling 33 (2001) 517–532

MATHEMATICAL AND COMPUTER MODELLING

www.elsevier.nl/locate/mcm

Branch and Cut Methods for Network Optimization

L. CACCETTA AND S. P. HILL
School of Mathematics and Statistics, Curtin University of Technology
GPO Box U1987, Perth, WA 6845, Western Australia
(caccetta)(hillsp)@cs.curtin.edu.au

(Received December 1997; revised and accepted July 1999)

Abstract—Combinatorial optimization problems arising in network applications are usually computationally difficult. Typically, one needs to solve a large mixed integer linear programming problem. Over the past decade, the method of branch and cut has emerged as a powerful technique for solving such problems. In this paper, we focus on computationally difficult network optimization problems, highlighting the effectiveness of branch and cut methods.

Keywords—Branch and cut, Cutting planes, Combinatorial optimization, Mixed integer linear programming.

1. INTRODUCTION

A network is a system which involves the movement or flow of some commodity such as, information products, resources, and people. Computer networks, electronic circuits, communication networks, assembly production lines, pipeline networks, and transportation networks are just a few common examples. Usually, real world networks are complex and the problems arising from the design and operation of these networks are extremely challenging. These problems include, the design of minimal cost reliable networks; the design of integrated circuits; the design of data communication networks capable of supporting large scale on-line applications; the design of powerful multiprocessors for solving complex problems, (e.g., radar signal processing) in real time; determining optimal routes in a given network; determining the most efficient mine extraction sequences; and many more.

Many networks may be viewed as a system of components, some of which interact in some way or another. For example, a communications network is a collection of centres (telephone exchanges, computer terminals, etc.) some of which communicate directly; a traffic network is a collection of intersections with road links connecting them; a printed circuit board is a collection of terminals, some of which are connected by conductor paths. Such systems can be conveniently modelled by a graph in which the vertices represent the components and the edges the interaction between the components. The resulting graph gives us a structural model of the network being studied, displaying clearly the interactions between the various components of the network. Factors that influence the design of a network such as, message delay time; throughput;

This work has been supported by Australian Research Grant No. A49532661.

0895-7177/01/$ - see front matter Typeset by $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX
PII: S0895-7177(00)00258-2

message traffic density; reliability; equipment capacity (terminal speeds, transmission rates, etc.); and many more; can be incorporated into the graph theoretic model by assigning appropriate weights to the vertices and edges of the graph representing the network. The result is a weighted graph (directed graph, when orientation is a factor as is the case, for example, in a traffic network) model. In many applications, the problem that arises is to construct a network which satisfies certain requirements and which is optimal according to some criterion such as cost, output, or performance. The requirements of the network can sometimes be expressed in terms of graph theoretic properties, usually as bounds on certain graph parameters. Thus, the problem usually becomes a graph optimization problem that can be formulated as a mixed integer linear programming problem.

In this paper, we focus on computationally difficult network optimization problems and we highlight the effectiveness of branch and cut methods to solve them. The use of branch and cut methods (and the recently introduced related branch and Price method [1]) have, over the past decade, emerged as extremely powerful techniques for solving large combinatorial optimization problems.

For convenience, we consider only networks involving undirected graphs; the case of directed networks is similar. In a graph optimization problem, one typically needs to choose a subset of E such that the resulting graph has the desired properties and is optimal with respect to an objective function that usually includes edge weights. For our purposes, $G = (V, E)$ denotes a finite graph with vertex set V and edge set E. Further, $V = \{1, 2, \ldots, n\}$ and thus $E = \{(i, j) : i, j \in V, i < j\}$. Note that we can assume without loss of generality that G is complete. For convenience, we denote the (undirected) edge between i and j by ij or ji. The weight (cost) of edge ij is denoted by c_{ij}. We use 0,1 decision variable x_{ij}s in our formulations; $x_{ij} = 1$ if edge ij is chosen and $x_{ij} = 0$ otherwise.

As an example of an integer linear programming problem, we consider the classic travelling salesman problem (TSP) which arises in many applications including production scheduling, network design, vehicle routing, crystallography, etc. In this problem, a salesman needs to visit, starting at the home town T_1, $n-1$ neighbouring towns $T_2, T_3, \ldots, T_n$ exactly once, and then returning home. Given the distance (cost) matrix $C = [c_{ij}]$, with c_{ij} being the distance between T_i and T_j, the objective is to find a tour that minimizes the total distance travelled. Recall that here $c_{ij} = c_{ji}$. Let $G = (V, E)$ be the graph representing the network. This problem can be formulated as follows:

$$\text{minimize } f = \sum_{i=1}^{n} \sum_{i<j} c_{ij} x_{ij}, \tag{1.1}$$

subject to

$$\sum_{j<i} x_{ji} + \sum_{j>i} x_{ij} = 2, \qquad 1 \le i \le n, \tag{1.2}$$

$$\sum_{i,j \in S} x_{ij} \le |S| - 1, \; S \subset V, \qquad 3 \le |S| \le n-2, \tag{1.3}$$

and

$$x_{ij} = 0, 1, \qquad \text{for all } i, j. \tag{1.4}$$

In the above, constraints (1.2) ensure that every town is visited and departed from exactly once. The subtour elimination constraints (1.3) ensure that the salesman's route is specified by exactly one tour.

In Section 3, we highlight some applications where branch and cut has proven effective. In particular, we deal with the TSP in Section 3.1. Other applications include a generalization of the TSP, the vehicle routing problem, detailed in Section 3.2, and an important problem in network design, namely the degree constrained minimum spanning tree problem is discussed in

Section 3.3. We conclude the paper with a brief discussion on general 0−1 problems in Section 3.4. For comprehensive surveys on integer programming, cutting planes, and branch and cut methods, we refer to [2–6].

In the following section, we present a detailed account of the branch and cut method. Our treatment includes the description of a general algorithm and a discussion of the method's advantages and disadvantages.

2. BRANCH AND CUT METHODS

A Linear Program (LP) may be expressed as

$$\begin{aligned} &\text{minimize} && f = c^{\top}x, \\ &\text{subject to} && Ax \leq b \text{ and } x \geq 0. \end{aligned} \tag{2.1}$$

If, in addition, some of the variables are specified as integer, then the problem is a mixed integer linear program (MILP). Of course, MILPs are far more difficult to solve than LPs. In the following discussion, we assume that some of the variables in (2.1) are integral. To solve the MILP problem usually requires the solution of one or more easier subproblems, where the integer restrictions or some of the other constraints are dropped. These easier problems are known as relaxations. A solution to a relaxed problem gives a lower bound to the original problem. These lower bounds can be used in a search tree technique to specify additional restrictions, and thus, obtain optimality. We can solve a problem of form (2.1) by subdividing its feasibility solution set into successively smaller subsets, placing bounds on the objective function value over each subset, and using these bounds to discard certain subsets from further consideration. This is the essence of the branch and bound search technique. The most common branch and bound search procedures are depth-first search and breadth-first searches (see [7]). Note that it is not usually possible to completely enumerate the search tree to get the optimal solution, due to the large number of possible solutions that need to be considered. Thus, the use of this approach is highly dependent upon the quality of the lower bounds that can be generated. High quality bounds can often be obtained using cutting plane procedures and constraints expressing the structure of the solution space. The method of branch and cut refers to algorithms using a cutting plane procedure to strengthen the lower bounds at each node within a branch and bound search routine. In addition, a large subset of constraints used are valid at each node of the search tree and this property is exploited.

Lower bounds can be obtained using the following general cutting plane procedure. Let K be a set of valid inequalities for Problem 2.1. Consider the problem

$$\begin{aligned} &\text{minimize} && f = c^{\top}x, \\ &\text{subject to} && Bx \leq d \text{ and } x \geq 0. \end{aligned} \tag{2.2}$$

Here, the constraints $Bx \leq d$ are a subset of the original constraints $Ax \leq b$. Also, $\{Ax \leq b\} \setminus \{Bx \leq d\}$ is a subset of K. A lower bound for (2.1) is generated using the following procedure adapted from Padberg and Rinaldi [8]. The procedure was pioneered by Dantzig *et al.* [9].

Cutting Plane Procedure

STEP 1. Set $\mathcal{L} = \phi$.

STEP 2. Solve Problem 2.2 and let $\bar{x}$ be its solution.

STEP 3. Find one or more inequalities in K that are violated by $\bar{x}$.

STEP 4. If none is found, stop. Otherwise, add the violating inequalities to $\mathcal{L}$ and go to Step 2.

The success of the above procedure is highly dependent upon efficiently finding sets of "strong" inequalities of K. The problem of finding a violating inequality of K or proving that no such

inequality exists for solution $\bar{x}$ is commonly referred to as the "separation problem". Ideally, an efficient exact method to solve the separation problem (Step 3) is required. Unfortunately, such methods are usually unavailable or are computationally expensive and heuristic procedures are employed.

To assist in the discussion of "strong" inequalities, we introduce some basic polyhedral theory terminology [10]. For $S \subseteq \mathbf{R}^n$, we let conv(S) denote the convex hull of S. An important result is that for finite S, conv(S) can be described by a finite set of linear inequalities. Further,

$$\min\{c^\top x : x \in S\} = \min\{c^\top x : x \in \text{conv}(S)\}.$$

Thus, any MILP can be represented as an LP, provided we know a set of linear inequalities that represent the solution space. Note that such a system of inequalities is usually incredibly large in number and generally unknown. To overcome these problems, the approach is to use a subset of the constraints defining conv(S) and/or constraints which are redundant in a minimal representation. For an MILP, a linear constraint that does not exclude any integer feasible points is called a cutting plane. If $\pi x \leq \pi_0$ is a valid inequality for $P = \{x \in \mathbf{R}^n : Ax \leq b\}$, and $F = \{x \in P : \pi x = \pi_0\}$, then F is called a face of P. A face of P is a facet of P if $\dim(F) = \dim(P) - 1$. This leads to the result that for each facet F of P, one of the inequalities representing F is necessary in the description of P. Thus, the use of facets in the description of the solution space yields a system of inequalities of smallest number. Also, if P defines the convex hull of integer solutions of a discrete optimization problem, then the use of facet defining inequalities is most likely to give the tightest lower bounds in the cutting plane procedure.

The above cutting plane procedure terminates when no further violations can be found or an optimal solution has been found. If an optimal solution is not obtained, then it is necessary to subdivide the solution space. To do this the cutting plane procedure is embedded in a search tree technique. In this case, a check that the lower bound generated at Step 2 is less than the best upper bound must be included. If the lower bound is at least equal to the best known upper bound, then the subproblem is fathomed. Step 4 is modified so that an integral solution which satisfies all inequalities of K becomes the best known solution and the upper bound is set accordingly. Note that if the above procedure is terminated when further violations can be found (which in some cases may be useful), then the objective function value is still a valid lower bound.

So that the LP basis does not become too large, constraints of the LP are typically removed when they are found to no longer be effective, i.e., the slack variable is nonbasic. This is either done as soon as the slack variable is found to be nonbasic or after the slack variable has been nonbasic for a prescribed number of iterations. Note, this operation does not alter the feasibility nor the optimality of the LP.

In a standard branch and bound method using an LP relaxation, the constraints present in the LP and the basis need to be stored at each child node. If the child node later needs to be explored, then the LP tableau is recreated. This process is expensive both in terms of time and memory requirements. The method of branch and cut overcomes this difficulty by using many constraints that are valid throughout the search tree, and so the LP need not be solved from scratch.

Note that child nodes usually differ in terms of the upper and lower bounds that have been set on the variables. Since the constraint set is usually valid throughout the search tree, the LP formed by setting bounds on variables and using any valid constraints currently present in the LP, can be solved to give a valid lower bound of the subproblem. The approach does not store the constraints and basis at the child nodes but rather, when resolving a child node, the lower bound generation procedure is initialized (Step 1 above) with any constraints of set K currently present in the LP, in addition to the constraints $Bx \leq d$. The current basis is also used when solving the LP for the first time at Step 2.

The Advantages and Disadvantages of the Method

- By using the constraints present in the LP from previous lower bound generations, large savings are made in terms of time and memory allocation, as constraints do not need to be stored or LP tableaus regenerated.
- Any valid cutting planes present in the LP from previous lower bound generations may be used to initialize the constraint set when generating the current lower bound. This means that the enumeration process does not need to be started from scratch each time. Further, when solving a node, the description of the solution polytope is being improved for all nodes of the search tree at the same time.
- Since constraints are removed from the LP tableau, they may need to be regenerated later in the search tree. Also, constraints may be removed too early in which case the lower bound may not be as high as it otherwise may have been. The advantage of removing constraints is that the size of the LP basis is reduced, and thus, the solution time and memory requirements are greatly decreased.
- If a node is explored which has vastly different restrictions to the node which was previously explored, then many of the constraints in the LP may not be tight and the initial lower bound may be poor. To overcome this problem with their breadth-first search method, Padberg and Rinaldi [8] keep a list of parent nodes and process the parent node before generating bounds for each of the child nodes. In addition, they keep a "pool" of previously generated constraints. This "pool" allows the regeneration of the parent node solution to be quite fast since less work needs to be done in finding violating constraints. Note, the above problem is not significant if depth-first search is used. Using this search method implies that the restrictions defining subproblems are similar from one child node to the next and therefore the constraints in the LP are likely to remain tight.
- When a child a node is removed from the list and the lower bound is generated, there is no guarantee that the generated lower bound will be at least equal to the lower bound value stored when the child node was placed on the list. This is because the constraints in the LP are probably different from when the node was stored. However, this will not be the usual case, as the additional restrictions placed on a child node, compared to that of its parent, usually results in a higher bound.
- For large problems, the number of variables in the LP relaxations may become sufficiently large, so as to make the LP solution time prohibitive, and thus, reduce the effectiveness of the branch and cut procedure. Consequently, reducing the number of $0-1$ variables through preprocessing is often required. Sparse graph techniques as used by Grötschel and Holland [11] may also reduce the computing times.
- As further research produces a better description of a problem's solution polytope, the use of the additional constraints, and methods for their detection may lead to violations being found more frequently at Step 3 of the above algorithm. As would be expected, if violations are detected more often, then the lower bounds usually improve. Thus, the method can be updated as further research advances are made. This update may not be possible if another relaxation is used, where typically one characteristic of the underlying structure is exploited to get a lower bound.

In recent years, branch and cut methods have been used in making large scale improvements in the size of the combinatorial optimization problems that can be solved exactly. Improvements have mainly been due to the following factors:

- the discovery of sets of constraints which specify feasibility and which frequently violate partial solutions (these constraints are often known to be facets);
- the discovery of methods, some of which are exact, for the quick detection of violations;
- recent technological improvements that have made available the computational power to run these algorithms.

In the next section, we discuss some examples where the branch and cut procedure has been effectively used.

3. APPLICATIONS

In this section, we discuss the use of branch and cut methods in solving MILPs. We consider the classical TSP and its close relative the vehicle routing problem (VRP). We also consider an MILP problem arising in network design. In all of these problems, the structure of the solution space is exploited using specialized cutting planes. We conclude the section with a discussion of general MILPs where the solution space structure is not immediately apparent.

3.1. The Travelling Salesman Problem

The TSP has been the classical testing problem for most combinatorial optimization techniques. Dantzig *et al.* [9] solved a 49-city problem in 1954 using an interactive procedure where cuts are identified and added manually. This early work motivated subsequent research in cutting plane procedures. In the early 1970s, a number of authors (see the survey [7]) tackled the TSP by considering a Lagrangian relaxation procedure which relaxes the degree constraints (1.2). The minimum weight spanning tree problem is the relaxed subproblem with the Lagrangian multipliers revised using subgradient optimization. This approach does not yield the most powerful method and has been superseded by cutting plane approaches.

Early results on the symmetric TSP polytope emanated from the work of Edmonds [12]. More specifically, a 2-matching (perfect 2-matching) of a graph is a set of edges such that every vertex is incident to at most (exactly) two edges. Note that every tour is a perfect 2-matching. Edmonds [12] gave a complete linear description of the 2-matching (perfect 2-matching) polytope. This work was further developed through a series of contributions from many authors. Amongst others, significant results were obtained by Chvátal [13], Grötschel and Padberg [14], and Grötschel and Pulleybank [15]. Constraints that are currently used in branch and cut algorithms were available by 1986. Though additional constraints such as regular star, chain, and ladder inequalities are facet defining, they have not yet been successfully incorporated into current computational methods; see, for example, [16–18].

In the following, we discuss those constraints that are used in computational procedures. For the TSP (1.1)–(1.4), the initial LP relaxation is

$$\begin{aligned} &\text{minimize} && f = \sum_{i=1}^{n} \sum_{i<j} c_{ij} x_{ij}, \\ &\text{subject to} && \sum_{j<i} x_{ji} + \sum_{j>i} x_{ij} = 2, \qquad 1 \le i \le n, \\ &\text{and} && 0 \le x_{ij} \le 1, \qquad \text{for all } i, j. \end{aligned}$$

This provides the initial lower bound on the optimal objective function value. This bound is improved by adding violating constraints (Step 3 of cutting plane procedure) of the following types.

Subtour elimination constraints (See [9].)

$$\sum_{i,j \in S} x_{ij} \le |S| - 1, \qquad S \subset V, \qquad 3 \le |S| \le n - 2. \tag{3.1}$$

Comb inequalities (See [14].)

Let $H, T_1, T_2, \ldots, T_k \subseteq V$, $k \geq 1$, be vertex sets satisfying

(i) $|H \cap T_i| \geq 1$, $1 \leq i \leq k$,
(ii) $|T_i \setminus H| \geq 1$, $1 \leq i \leq k$,
(iii) $T_i \cap T_j = \emptyset$, for $i \neq j$,
(iv) $k \geq 3$, and odd.

The comb inequalities are given by

$$\sum_{i,j \in H} x_{ij} + \sum_{p=1}^{k} \sum_{i,j \in T_p} x_{ij} \leq |H| + \sum_{i=1}^{k} |T_i| - \frac{1}{2}(3k+1). \tag{3.2}$$

Note, that for the special case $|T_i| = 2$ for each i, we have the 2-matching inequalities.

Clique tree inequalities (See [15].)

Let $H_1, H_2, \ldots, H_r$, $T_1, T_2, \ldots, T_s$, s odd, be vertex sets satisfying

(i) $|T_i \cap T_j| = 0$, for $i \neq j$,
(ii) $|H_i \cap H_j| = 0$, for $i \neq j$,
(iii) $2 \leq |T_i| \leq n-2$ and $|T_i \setminus \bigcup_{j=1}^{r} H_j| \geq 1$, for each i,
(iv) each H_i intersects an odd number (≥ 3) of T_is,
(v) the intersection graph of the sets H_is and T_js forms a tree.

Let t_i denote the number of H_js that T_i intersects. Then, the clique tree inequalities are given by

$$\sum_{p=1}^{r} \sum_{i,j \in H_p} x_{ij} + \sum_{q=1}^{s} \sum_{i,j \in T_q} x_{ij} \leq \sum_{p=1}^{r} |H_p| + \sum_{q=1}^{s} (|T_q| - t_q) - \frac{1}{2}(s+1). \tag{3.3}$$

Note that the set of subtour elimination constraints and comb inequalities are subsets of the clique tree inequalities.

Exact methods for finding subtour elimination constraints are available (see [19,20]). Further, exact methods for finding comb inequalities are known for the 2-matching constraints case [21]. For other cases, heuristics are used; often, for computational reasons, heuristics are used prior to or in place of exact methods. The usefulness of good heuristics for these separation problems is best demonstrated in [22,23].

Over the past two decades, there has been considerable improvement in the size of TSPs solved exactly; from a 120-city problem in 1977 to a 13,509-city problem in 1998. Grötschel [24] solved a 120-city problem using an interactive procedure. Crowder and Padberg [25] solved ten 318-city problems using heuristics to identify subtour elimination and comb constraint. Grötschel and Holland [11] solved a 1000-city problem using an exact method [19] for finding subtour elimination constraints as well as an exact method [21] for the 2-matching constraints. Other comb inequalities were found using heuristics. Padberg and Rinaldi [8,20,26] were the first to apply the branch and cut method to the TSP. Prior to this, the approach used was a cutting plane method followed by branch and bound. In their work, Padberg and Rinaldi also improved the efficiency for finding subtour elimination constraints and the 2-matching constraints. In addition, they used heuristics for finding clique tree inequalities. The largest problem solved was a 2,392-city problem.

Further improvements have recently been announced by Applegate *et al.* [22] who solved a 7,397-city problem and later a 13,509-city problem [23]. Some of the major improvements have come from new ways of finding sets of violating cuts as well as a parallel implementation of the algorithm using 48 UNIX workstations. Of note, Applegate *et al.* [22,23] efficiently use a "pool" of constraints as well as a "gluing" operation to find "clique tree like" inequalities which are a (large) subset of clique tree inequalities.

We conclude this section by noting that a "larger" problem does not necessarily imply a "more difficult" problem. For example, the Applegate *et al.* [22,23] algorithm is the only one to solve a 225-city problem which is contrived to be hard. All test problems are available in TSPLIB (see [27]).

3.2. Vehicle Routing Problem

Vehicle routing problems (VRPs) are concerned with the delivery of some commodities from one or more depots to a number of customer locations with known demand. Such problems arise in many physical systems dealing with distribution. For example, delivery of commodities such as mail, food, newspapers, etc. The specific problem which arises is dependent upon the type of constraints and management objective. The constraints of the problem arise from the vehicle capacity, distance/time restriction, number of customers serviced by a vehicle, and other practical requirements. The management objectives usually relate to the minimization of cost/distance or fleet size. Much of the literature on vehicle routing has been concerned with problems having the following features.

(i) A single commodity is to be distributed from a single depot to customers with known demand.
(ii) Each customer's demand is serviced by one vehicle.
(iii) Each vehicle has the same capacity and makes one trip.
(iv) Total distance travelled by each vehicle cannot exceed a specified limit.
(v) Each customer must be serviced within a specified time window.
(vi) The objective is to minimize the total distance travelled by all vehicles.

Relaxing Restrictions (iv) and (v) gives rise to the so-called capacitated vehicle routing problem (CVRP), the most studied VRP. We focus on this problem here.

The VRP was first mentioned by Garvin *et al.* [28] in relation to the distribution of gasoline to service stations, using a fleet of vehicles of various capacities. It was initially formulated by Dantzig and Ramser [29] as a $0-1$ integer problem, and a linear programming based heuristic was proposed. Much of the early work on the VRP focused on heuristics. Though heuristics still dominate, over the past 15 years or so, there has been steady progress in the development of useful exact methods. Early work using branch and cut techniques was carried out by Laporte *et al.* [30–35]. With these exact methods, many ideas have come from the extensive research of the TSP, since it is a related problem. Thus, many techniques (see [36]) utilize the tools of linear programming, dynamic programming, set partitioning, and Lagrangian relaxation. Branch and cut methods have proven to be superior for loosely constrained problems and competitive with set partitioning based methods for tightly constrained problems. The largest CVRP literature problem solved to date is a 134-city problem [37,38]; this was accomplished using branch and cut. Now, to the details.

We denote the depot by 1 and the set of customer locations by $C = \{2, 3, \ldots, n\}$. Thus, the graph $G = (V, E)$ representing the vehicle network has $V = \{1, 2, \ldots, n\}$ and $E = \{(i, j) : i, j \in V, i < j\}$. We adopt the following notation:

q_j	demand of customer j, $j \in C$,
c_{ij}	distance between locations i and j,
m	number of delivery vehicles,
Q	common vehicle capacity.

We assume throughout that $Q \geq \max\{q_j\}$, and the distance matrix $C = (c_{ij})$ is symmetric and its elements satisfy the triangle inequality. For $S \subseteq C$, we let $\ell(S) =$ a lower bound on the number of vehicles required to visit all locations of S in an optimal solution. Note that $\ell(S) \geq 1$. We

write $\bar{S}$ for the complement of C of S. For $i, j \in V$, our decision variables are defined as

$$x_{ij} = \begin{cases} 1, & \text{if a vehicle travels on a single trip between } i \text{ and } j, \\ 2, & \text{if } i = 1 \text{ and } (1, j, 1) \text{ is a route,} \\ 0, & \text{otherwise.} \end{cases}$$

Note that x_{ij} only needs to be defined for $i < j$, and $q_i + q_j \leq Q$. Laporte *et al.* [35] formulated the problem as follows:

$$\text{minimize} \quad f = \sum_{i<j} c_{ij} x_{ij}, \tag{3.4}$$

$$\text{subject to} \quad \sum_{j \in C} x_{1j} = 2m, \tag{3.5}$$

$$\sum_{i<k} x_{ik} + \sum_{j>k} x_{kj} = 2, \qquad \text{for each } k \in C, \tag{3.6}$$

$$\sum_{i,j \in S} x_{ij} \leq |S| - \ell(S), \qquad \text{for all } S \subseteq C \text{ with } 3 \leq |S| \leq n-2, \tag{3.7}$$

$$x_{ij} = \begin{cases} 0, 1, \text{ or } 2 & \text{for } i = 1, \\ 0, 1, & \text{otherwise,} \end{cases} \tag{3.8}$$

$$m \text{ is a positive integer.} \tag{3.9}$$

In the above formulation, m can be fixed or variable. Constraints (3.5) and (3.6) are degree constraints on the depot and the customer locations, respectively. Constraints (3.7) eliminate subtours, that is, they prohibit subtours that are free from the depot, or connected to the depot but violate the capacity or distance restrictions. In an actual implementation of the above formulation, $\ell(S)$ is only calculated when required and typically $\lceil (1/Q) \sum_{j \in S} q_j \rceil$ is used. In a recent publication, Augerat *et al.* [38] quoted a private communication where the separation problems for (3.7) using this value of $\ell(S)$ has been shown to be NP-complete. This indicates that the VRP is a much more difficult problem than the TSP. Violations of (3.7) using $\ell(S) = (1/Q) \sum_{j \in S} q_j$ were found exactly in [38]. Another approach for finding violations of (3.7) is to search for violations of equivalent constraints. Two computationally useful [39,40] constraints are

$$\sum_{i,j \in S} x_{ij} + \sum_{i \in S} x_{1i} - m \leq |S| - \ell(\bar{S}), \tag{3.10}$$

where $\bar{S} = C \setminus S$.

$$\sum_{j \in P} x_{1j} + \sum_{i,j \in P} x_{ij} + \sum_{i,j \in S} x_{ij} + \sum_{E[P,\bar{S}]} x_{ij} \leq |S| + |P| - \ell(\bar{P}), \tag{3.11}$$

where $P \subseteq S \subseteq C$, $\bar{P} = S \setminus P$, $\bar{S} = C \setminus S$, and $E[S,T] = \{(i,j) : i \in S, j \in T\}$.

Recently, motivated by the success of branch and cut methods for the TSP, many cutting planes have been developed for the CVRP. These include the following.

Comb inequalities

Let $W_0, W_1, \ldots, W_s \subseteq C$ satisfy

(1) $|W_i \setminus W_0| \geq 1$, $i = 1, \ldots, s$,
(2) $|W_i \cap W_0| \geq 1$, $i = 1, \ldots, s$,
(3) $|W_i \cap W_j| = 0$, $1 \leq i \leq j \leq s$, and
(4) $s \geq 3$ and odd.

The comb inequality is given by

$$\sum_{p=0}^{s} \sum_{i,j \in W_p} x_{ij} \leq \sum_{p=0}^{s} |W_p| - \frac{3s+1}{2} + \alpha(m-1), \tag{3.12}$$

where

$$\alpha = \begin{cases} 0, & \text{if } 1 \notin \bigcup_{i=0}^{s} W_i, \\ 1, & \text{if } 1 \in W_0 \setminus \bigcup_{i=1}^{s} W_i \text{ or } 1 \in W_j \setminus W_0, \text{ for some } j = 1, \ldots, s, \\ 2, & \text{if } 1 \in W_j \cap W_0, \text{ for some } j = 1, \ldots, s. \end{cases}$$

The case $\alpha = 0$ was initially addressed by Laporte and Norbert [34] and Laporte and Bourjolly [41]. Cornuéjols and Harche [42] show that the comb inequalities (3.11) may be strengthened if, in addition to Conditions 1–4, the comb satisfies $1 \in W_1 \setminus W_0$. Then, the following comb inequality holds:

$$\sum_{p=0}^{s} \sum_{i,j \in W_p} x_{ij} \leq \sum_{p=0}^{s} |W_p| - \frac{3s+1}{2} + m - \ell(C \setminus W_1). \tag{3.13}$$

Cornuéjols and Harche [42] use (3.7) and (3.12) with $\alpha = 0$ and (3.13) to solve an 18 customer problem, using three different values for m, and a 50 customer benchmark problem. Violating cuts were identified by hand and added to the LP relaxation.

Bin packing inequalities

The bin packing problem (BPP) has an objective of packing k items having positive weights $w_1, w_2, \ldots, w_k$ into the least number (say M) of bins $B_1, B_2, \ldots, B_m$ such that the total weight in each bin is at most Q. We denote this problem by BPP$\{w_1, w_2, \ldots, w_k; Q\}$.

Consider a set of customers $S \subseteq C$ and a partition $P_S = (P_1, \ldots, P_u)$ of S such that

$$\sum_{k \in P_i} q_k \leq Q, \qquad 1 \leq i \leq u.$$

Let $\hat{M}(P_S)$ be a lower bound to the minimum number of bins M^*, required for BPP $\{\hat{w}_1, \hat{w}_2, \ldots, \hat{w}_k; Q\}$ where $\hat{w}_i = \sum_{k \in P_i} q_k$. Then,

$$\sum_{i,j \in S} x_{ij} + \sum_{k=1}^{u} \sum_{i,j \in P_k} x_{ij} \leq 2|S| - u - \hat{M}(P_S) \tag{3.14}$$

is a valid inequality for the CVRP. This was established by Achuthan *et al.* [37] with related results having also been presented in [38].

Generalized connectivity constraints

Let $S, T_1, T_2, \ldots, T_k \subseteq C$ be such that

(1) $k \geq v > 0$ and $\sum_{i \in S} q_i + \sum_{j=1}^{v} \sum_{i \in T_{p_j}} q_i > \ell(S)Q$, with $1 \leq p_i \neq p_j \leq k$,
(2) $T_i \cap T_j = \phi$, $i \neq j$,
(3) $S \cap T_i = \phi$, $1 \leq i \leq k$, and
(4) $T = \bigcup_{i=1}^{k} T_i$.

Then, any feasible solution (x_{ij}) of the CVRP satisfies

$$A(S,v,k) \sum_{i,j \in S} x_{ij} + \sum_{E[S,T]} x_{ij} + \sum_{p=1}^{k} \sum_{i,j \in T_p} x_{ij} \leq A(S,v,k)(|S| - \ell(S)) + |T| - k + v - 1, \tag{3.15}$$

where

$$A(S,v,k) = \begin{cases} \min\{k-v+1, 2\ell(S)-v+3\}, & \text{if } 2\ell(S)-v+3>1, \\ k-v+1, & \text{otherwise.} \end{cases}$$

In addition, for $R \subseteq S$, any feasible solution of the CVRP satisfies

$$\begin{aligned} &(2\ell(R)+v-1)\sum_{i,j\in S} x_{ij} + 2\sum_{i,j\in R} x_{ij} + \sum_{E[R,T]} x_{ij} + \sum_{p=1}^{k}\sum_{i,j\in T_p} x_{ij} \\ &\leq (2\ell(R)+v-1)(|S|-\ell(S)) + 2(|R|-\ell(R)) + |T| - k + v - 1. \end{aligned} \tag{3.16}$$

Equation (3.15) strengthens and generalizes similar constraints presented by Fisher [43]. Both (3.15) and (3.16) were established in [37].

Optimality based constraints

All the cutting planes presented above are based on feasibility requirements. Recently, Achuthan *et al.* [40] developed new cutting planes based on a specified structure of an optimal solution. Any feasible solution which does not satisfy these cutting planes need not be considered whilst searching for an optimal solution. Thus, these cutting planes will, in effect, reduce the number of feasible solutions that need to be considered. The inequalities are the following.

- There exists an optimal solution $X = (x_{ij})$ of the CVRP (3.4) to (3.9) with variable m satisfying the following set of constraints:

$$\sum_{i,j\in S} x_{ij} + \sum_{j\in S} x_{1j} \leq |S| + 1, \tag{3.17}$$

  for all $S \subseteq C$ with $2 \leq |S| \leq |C|$ and $\sum_{i\in S} q_i \leq Q$.
- There exists an optimal solution $X = (x_{ij})$ of the CVRP (3.4) to (3.9) with variable m satisfying (3.17) and the following set of constraints:

$$\sum_{i,j\in S} x_{ij} + \sum_{i\in S} x_{1i} \leq |S| + \left\lfloor \frac{2\left(\sum_{i\in S} q_i + \delta\right)}{(Q+1+\delta)} \right\rfloor, \tag{3.18}$$

  for all $S \subseteq C$ with $2 \leq |S| \leq |C|$ and $\sum_{i\in S} q_i > Q$. $\delta = 0, 1$ according as Q is odd or even.

From inequalities (3.17) and (3.18), one can obtain the only available nontrivial upper bound for m.

In a series of papers, Achuthan *et al.* [37,39,40] developed a branch and cut algorithm incorporating (3.4)–(3.11) and (3.14)–(3.18). The algorithm was tested on 1650 simulated test problems with size of up to 100 customers plus 14 benchmark literature problems with size up to 199 customers. The largest problem solved to optimality had 134 customers.

Augerat *et al.* [38] have also recently described a branch and cut algorithm for the CVRP. Features of this method include extensive searching for violations of the subtour elimination constraints, the use of comb inequalities, the use of constraints incorporating a BPP lower bound similar to (3.14), and the use of "hypotour inequalities". Also, they use a sophisticated branching rule to branch on sets of customers. This branching rule incorporates ideas used by Applegate *et al.* [22] for the TSP and Fisher [43] for the CVRP. They too solve the 134-customer problem.

Preliminary analysis indicates that the initial lower bounds (root node) of Augerat *et al.* [38] are slightly better than those of Achuthan *et al.* [37]. However, they do take much longer to generate. An advantage of using the method of branch and cut is that research into the description of the solution polytope, as well as improved methods for the detection of violating constraints can lead to updates in the method. Thus, we would expect to see a highly competitive algorithm produced by the marriage of these two methods. It seems likely that the algorithm would also benefit from the utilisation of the untested generalised multistar constraints presented by Achuthan *et al.* [37]; these constraints generalise those presented by Araque *et al.* [44].

3.3. Degree Constrained Minimum Spanning Tree Problem

In many network applications, a graph theoretic problem that frequently arises is that of finding, in a given weighted graph, a minimum weight tree satisfying certain properties (see [45]). Here, we consider the case when there are degree restrictions on the vertices of the spanning tree. The resulting degree constrained minimum spanning tree problem (DCMST) arises naturally in communication networks where the degree restriction on a vertex may represent the number of line interfaces available at a terminal or the traffic capacity of the terminal.

The DCMST problem has been considered by a number of authors and both heuristic and exact methods have been proposed. A comprehensive account of heuristics is given in [46]. Exact methods have been tested for the case of identical degree restrictions by Narula and Ho [47], Gavish [48], Savelsbergh and Volgenant [49] and Volgenant [50]. The testing has been carried on symmetric problems. Euclidean problems have been observed to be much easier than non-Euclidean [49]. For the Euclidean case, Narula and Ho [47] solved, using branch and bound, problems up to 100 vertices, Gavish [48] solved, using Lagrangian relaxation, problems with up to 200 vertices, and Savelsbergh and Volgenant [49] solved, using branch and bound, problems with up to 200 vertices. For the non-Euclidean case, Savelsbergh and Volgenant [49] solved using branch and bound, problems with up to 70 vertices and Volgenant [50] solved, using Lagrangian relaxation, problems with up to 150 vertices. Recently [51], we described a branch and cut method that solves problems with up to 800 vertices. The method was tested on 3,150 random table problems ranging from 100 to 800 vertices.

In this work, due to the size of the MILPs, the method is computationally feasible only if the number of $0-1$ variables can be drastically reduced. This is accomplished by using a version of the Lagrangian relaxation method of Volgenant [50] in a preprocessing procedure. Volgenant [50] demonstrated that by using tight upper and lower bounds, a large percentage of edges could be fixed as included or excluded in the optimal spanning tree.

Denoting the upper bound on the degree of vertex i by b_i and the degree of vertex i in the spanning tree by d_i, an MILP formulation of the DCMST problem is

$$\text{minimize} \quad \sum_{i,j \in V} c_{ij} x_{ij}, \tag{3.19}$$

$$\text{subject to} \quad \sum_{i,j \in V} x_{ij} - d_i = 0, \qquad 1 \le i \le n, \tag{3.20}$$

$$\sum_{i=1}^{n} d_i = 2(n-1), \tag{3.21}$$

$$\sum_{i,j \in S} x_{ij} \le |S| - 1, \qquad \text{for all } \phi \ne S \subset V, \tag{3.22}$$

$$1 \le d_i \le b_i, \qquad 1 \le i \le n, \qquad \text{and} \tag{3.23}$$

$$x_{ij} = 0, \qquad \text{or } 1. \tag{3.24}$$

The initial relaxation is (3.19)–(3.21) and (3.23) together with $0 \le x_{ij} \le 1$ for all i, j. Violating cuts of type (3.22) and the connectivity constraints

$$\sum_{E[S,\bar{S}]} x_{ij} \ge 1, \qquad \text{for all } \phi \ne S \subset V, \tag{3.25}$$

are used to improve the lower bounds.

3.4. General $0-1$ Problems

In most applications, cutting plane sets are discovered by first noting some structural aspect of the feasible or optimal solution space, and then expressing this structure in terms of linear

constraints. When such structural properties are not immediately apparent, other strategies must be used in the generation of cutting places.

In the early days of integer programming, Gomory cuts [19,52–54] generated considerable interest as it was believed that they could be utilized to efficiently solve integer problems. These fell out of favour because of the poor convergence. In modern terminology, Gomory cuts are typically not facet defining nor valid at each child node of the search tree. Much of the research effort, over the past twenty years or so, on cutting place procedures has focused on developing cuts that do not suffer from these drawbacks. However, recently Balas *et al.* [55] rekindled interest in Gomory cuts. They solved that by using a simple "lifting procedure", Gomory cuts can be utilized throughout the search tree of $0-1$ MILPs. Furthermore, these cuts prove computationally useful if they are generated in sets. However, they do not incorporate these cuts into their "MIPO" [56] routine.

The first computational results for large scale $0-1$ linear programs was presented by Crowder *et al.* [57] in 1983. Their methods utilized simple preprocessing procedures to strengthen the initial formulation. Two sets of cutting planes are then used to strengthen the solution at the root node. If needed, branch and bound is used to find an optimal solution, with standard reduced cost fixing used at the child nodes to eliminate variables. Because the method draws on results developed from the study of the knapsack polytope, it is most effective for sparse matrices where the constraint coefficients are not plus or minus 1. Van Roy and Wolsey [58] utilise a similar approach for mixed $0-1$ problems.

Boyd [59] developed a cutting plane algorithm that uses Fenchel cuts which are based on "Fenchel duality". To strengthen the solution, cuts are generated by maximising a piecewise linear function over a nonlinear domain. Since this second problem is difficult to solve, heuristics and different relaxations of the domain are used. The problems of [57] are used as the test set. Computational results indicate that Fenchel cuts appear to be as useful as the constraints used by Crowder *et al.* [57].

Techniques for strengthening the linear programming relaxation of an integer linear program (ILP) have focused recently on lifting the problem into a higher-dimensional space to get a tighter relaxation and then projecting the formulation back onto the original space. Sherali and Adams [60] and Lovász and Schrijver [61] proposed methods that roughly require the squaring of the number of variables and constraints to obtain the convex hull of feasible ILP solutions. Balas *et al.* [62] further develop this technique and show that the convex hull may be obtained by doubling, rather than squaring, the number of variables and constraints.

To use these results in a practical application, Balas *et al.* [56] generate facet defining cuts by solving a second linear program which is about twice as large as the original. These cuts have essentially the same properties as those derived theoretically, but are computationally cheaper to generate. This is done by working in the subspace defined by the fractional solution variables, and then lifting and strengthening the generated inequality. This algorithm is demonstrated to be superior in most cases to well-known commercially available MILP solution routines.

Recently Boyd [63] developed a cutting plane algorithm whose complexity is unrelated to the number of facets defining the underlying solution space polyhedron. The algorithm uses properties of the ellipsoid algorithm, an "interior point" method for solving linear programs. In theory, the ellipsoid algorithm should take less time to solve linear programs than the simplex method, though in practice, this has not been the case with the algorithm performing poorly. Thus, if implemented, the new cutting plane algorithm would be expected to suffer from the same drawbacks, and not be useful in practice. An interesting result of this paper is to show that the cardinality versions of many combinatorial optimisation problems are solvable in polynomial time using the proposed method. In particular, it is demonstrated that for problems containing a ball of polynomially bounded radius, the presented algorithm is polynomial in the running time in the solution of the separation problem and pseudo-polynomial in the size of the objective function.

4. CONCLUSION

The method of branch and cut draws on many areas in combinatorial optimization. Thus, any advances in a range of fields and for different problems may result in significant reductions in solution times for an unrelated problem. In particular, we can expect to see reductions in solution times and larger problems being solved through improvements in such areas as LP solution times, heuristic techniques, preprocessing and reformulation of the problems, methods for reducing the size of the search tree, development of exact and heuristic methods for solving separation problems, and parallelisation and the discovery of new cutting planes. Since this is the case, we can be assured that branch and cut will remain the method of choice for many combinatorial optimisation problems and an active field of research for many years to come.

REFERENCES

1. M. Savelsbergh, A branch-and-price algorithm for the generalized assignment problem, *Operations Research* (to appear).
2. K. Aardal and C.P.M. Van Hoesel, Polyhedral techniques in combinatorial optimization I: Theory, *Statistica Nederlandica* **50**, 4–26 (1996).
3. K. Aardal and C.P.M. Van Hoesel, Polyhedral techniques in combinatorial optimization II: Computations, *Statistica Nederlandica* (to appear).
4. A. Caprara and M. Fischetti, Branch-and-cut algorithms, In *Annotated Bibliographies in Combinatorial Optimization*, (Edited by M. Dell'Amico, F. Maffioli and S. Martello), J. Wiley and Sons, Chichester, (1997).
5. M. Jünger, G. Reinelt and S. Thienel, Practical problem solving with cutting plane algorithms in combinatorial optimization, Combinatorial optimization, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, (Edited by W. Cook, L. Lovász and P. Seymour), pp. 111–152, American Mathematical Society, (1995).
6. A. Lucena and J.E. Beasley, Branch and cut algorithms, In *Advances in Linear and Integer Programming*, (Edited by J.E. Beasley), pp. 187–221, Oxford University Press, (1996).
7. E. Balas and P. Toth, Branch and bound methods, In *The Travelling Salesman Problem*, (Edited by E.L. Lawler, J.K. Lenstra, A.G.H. Rinnooy Kan and D.B. Shmoys), pp. 361–401, John Wiley and Sons, 361–401, (1985).
8. M. Padberg and G. Rinaldi, A branch and cut algorithm for the resolution of large scale travelling salesman problems, *SIAM Review* **33** (1), 60–100 (1991).
9. G.B. Dantzig, D.R. Fulkerson and S.M. Johnson, Solution of a large scale travelling salesman problem, *Operations Research* **2**, 393–410 (1954).
10. G.L. Nemhauser and L.A. Wolsey, *Integer and Combinatorial Optimization*, John Wiley and Sons, (1988).
11. M. Grötschel and O. Holland, Solution of large-scale symmetric travelling salesman problems, *Mathematical Programming* **51**, 141–202 (1991).
12. J. Edmonds, Maximum matching and a polyhedron with 0,1 vertices, *Journal of Research of the National Bureau of Standards* **69B**, 67–72 (1965).
13. V. Chvátal, Edmonds polytopes and weakly Hamiltonian graphs, *Math. Programming* **5**, 29–40 (1973).
14. M. Grötschel and M.W. Padberg, Polyhedral theory, In *The Travelling Salesman Problem*, (Edited by E.L. Lawler, J.K. Lenstra, A.G.H. Rinnooy Kan and D.B. Shmoys), pp. 251–305, John Wiley and Sons, (1985).
15. M. Grötschel and W.R. Pulleybank, Clique tree inequalities and the symmetric travelling salesman problem, *Mathematics of Operations Research* **11**, 537–569 (1986).
16. G. Cornuéjols, F. Fonlupt and D. Naddef, The travelling salesman problem on a graph and some related integer polyhedra, *Mathematical Programming* **33**, 1–27 (1985).
17. B. Fleishmann, A new class of cutting planes for the symmetric travelling salesman problem, *Mathematical Programming* **40**, 225–246 (1988).
18. S.T. Boyd and W. Cunningham, Small travelling salesman polytopes, *Math. Operations Research* **16**, 259–271 (1991).
19. R.E. Gomory and T.C. Hu, Multi-terminal network flows, *SIAM Journal on Applied Mathematics* **9**, 551–570 (1961).
20. M. Padberg and G. Rinaldi, An efficient algorithm for the minimum capacity cut problem, *Mathematical Programming* **47**, 19–36 (1990).
21. M.W. Padberg and M.R. Rao, Odd minimum cut sets and b-matchings, *Mathematics of Operations Research* **7**, 67–80 (1982).
22. D. Applegate, R. Bixby, V. Chvátal and W. Cook, Finding cuts in the TSP (a preliminary report), DIMACS Technical Report (1995).
23. D. Applegate, R. Bixby, V. Chvátal and W. Cook, On the solution of travelling salesman problems, *Documenta Mathematica (Journal der Deutschen Mathematiker-Vereinigung)*, International Congress of Mathematicians, 645–656 (1998).

24. M. Grotschel, On the symmetric travelling salesman problem: Solution of a 120-city problem, *Mathematical Programming Studies* **12**, 61–77 (1980).
25. H. Crowder and M.W. Padberg, Solving large-scale symmetric travelling salesman problems to optimality, *Management Science* **26**, 495–509 (1980).
26. M. Padberg and G. Rinaldi, Facet identification for the symmetric travelling salesman polytope, *Mathematical Programming* **47**, 219–257 (1990).
27. G. Reinelt, A travelling salesman problem library, *ORSA Journal of Computing* **3**, 376–384 (1991).
28. W.M. Garvin, H.W. Crandall, J.B. John and R.A. Spellman, Applications of linear programming in the oil industry, *Management Science* **3**, 407–430 (1957).
29. G.B. Dantzig and J.H. Ramser, The truck despatching problem, *Management Science* **6**, 80–91 (1959).
30. G. Laporte, M. Desrochers and Y. Nobert, Two exact algorithms for the distance constrained vehicle routing problem, *Networks* **14**, 161–172 (1984).
31. G. Laporte, H. Mercure and Y. Nobert, An exact algorithm for the asymmetrical capacitated vehicle routing problem, *Networks* **16**, 33–46 (1986).
32. G. Laporte, H. Mercure and Y. Nobert, A branch and bound algorithm for a class of asymmetrical vehicle routing problems, *Journal of the Operational Research Society* **43**, 469–481 (1992).
33. G. Laporte and Y. Nobert, A cutting plane algorithm for the m-salesman problem, *Journal of the Operational Research Society* **31**, 1017–1023 (1980).
34. G. Laporte and Y. Nobert, Comb inequalities for the vehicle routing problem, *Methods of Operational Research* **51**, 271–276 (1984).
35. G. Laporte, Y. Nobert and M. Desrochers, Optimal routing under capacity and distance restrictions, *Operations Research* **33**, 1050–1073 (1985).
36. G. Laporte, The vehicle routing problem: An overview of exact and approximate algorithms, *European Journal of Operational Research* **59**, 345–358 (1992).
37. N.R. Achuthan, L. Caccetta and S.P. Hill, The capacitated vehicle routing problem: Some new cutting planes, *Asia Pacific Journal of Operational Research* **15**, 109–123 (1998).
38. P. Augerat, J.M. Belengeur, E. Benavent, A. Corberan, N. Naddef and G. Rinaldi, Computational results with a branch and cut code for the capacitated vehicle routing problem, Research Report 949-M, Universite Joseph Fourier, Grenoble, France (1995).
39. N.R. Achuthan, L. Caccetta and S.P. Hill, A new subtour elimination constraint for the vehicle routing problem, *European Journal of Operations Research* **91**, 573–586 (1995).
40. N.R. Achuthan, L. Caccetta and S.P. Hill, An improved branch and cut algorithm for the capacitated vehicle routing problem Technical Report, Curtin University of Technology, Western Australia (1996).
41. G. Laporte and J.M. Bourjolly, Some further results on K-star constraints and comb inequalities, *Cahiers Du GERAD, G-82-10, Ecole des Hautes Etudes Commerciales de Montréal* (1984).
42. G. Cornuéjols and F. Harche, Polyhedral study of the capacitied vehicle routing problem, *Mathematical Programming* **60**, 21–52 (1993).
43. M. Fisher, Optimal solution of vehicle routing problems using minimum K-trees, *Operations Research* **42**, 626–642 (1994).
44. J.R. Araque, G. Kudva, T.L. Morin and J.F. Pekny, A branch-and-cut algorithm for vehicle routing problems, *Annals of Operations Research* **50**, 37–59 (1994).
45. L. Caccetta, Graph theory in network design and analysis, In *Recent Studies in Graph Theory*, (Edited by V.R. Kulli), pp. 29–63, Vishwa International, (1989).
46. L. Caccetta, B.K. Gan and S.P. Hill, Heuristics for the degree constrained restricted spanning tree problem, Technical Report, Curtain University of Technology, Western Australia, (1997).
47. S.C. Narula and C.A. Ho, Degree-constrained minimum spanning tree, *Computers and Operations Research* **33**, 1050–1073 (1985).
48. B. Gavish, Topological designs of centralized computer networks—Formulation and algorithms, *Networks* **12**, 355–377 (1982).
49. M. Savelsbergh and A. Volgenant, Edge exchange in the degree-constrained minimum spanning tree problem, *Computers and Operations Research* **12**, 341–368 (1985).
50. A. Volgenant, A Lagrangian approach to the degree-constrained minimum spanning tree problem, *European Journal of Operations Research* **39**, 325–331 (1989).
51. L. Caccetta and S.P. Hill, A branch and cut method for the degree constrained minimum spanning tree problem, *Networks* (to appear).
52. R.E. Gomory, Outline of an algorithm for integer solutions to linear programs, *Bulletin of the American Mathematical Society* **64**, 275–278 (1958).
53. R.E. Gomory, Solving linear programming problems in integers, In *Proceedings of the Symposium on Applied Mathematics*, Volume 10, pp. 211–215, (1960).
54. R.E. Gomory, An algorithm for integer solutions to linear programs, In *Recent Advances in Mathematical Programming*, (Edited by R.L. Graves and P. Wolfe), pp. 269–302, McGraw-Hill, New York, (1963).
55. E. Balas, S. Ceria, G. Cornuejols and N Natraj, Gomory cuts revisited, *Operations Research Letters* **19**, 1–9 (1996).
56. E. Balas, S. Ceria and G. Cornuejols, Mixed 0-1 programming by list-and-project in a branch-and-cut framework, *Management Science* **42**, 1229–1246 (1996).

57. H. Crowder, E.L. Johnson and M. Padberg, Solving large-scale zero one linear programming problems, *Oper. Res.* **31**, 803–834 (1983).
58. T.J. Van Roy and L.A. Wolsey, Solving mixed integer programming problems using automatic reformation, *Operations Research* **35**, 45–57 (1987).
59. E.A. Boyd, Fenchel cutting planes for integer programs, *Operations Research* **42**, 53–64 (1994).
60. H. Sherali and W. Adams, A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems, *SIAM Journal of Discrete Math.* **3**, 411–430 (1990).
61. L. Lovász and A. Schrijver, Cones of matrices and set-functions of 0-1 optimization, *SIAM Journal on Optimization* **1**, 161–190 (1991).
62. E. Balas, S. Ceria and G. Cornuejols, A lift-and-project cutting plane algorithm for mixed 0-1 programs, *Mathematical Programming* **58**, 295–324 (1993).
63. E.A. Boyd, On the complexity of a cutting plane algorithm for solving combinatorial linear programs, *SIAM Journal on Discrete Mathematics*, 365–376 (1996).