A relax and cut algorithm for the vehicle routing problem

Carlos Martinhon
Departamento de Ciência da Computação, Instituto de Computação
Universidade Federal Fluminense
Rua Passo da Pátria 156, São Domingos, Niteroi, RJ, 24210-240, Brazil
mart@dcc.ic.uff.br

Abilio Lucena
Departamento de Administração
Universidade Federal do Rio de Janeiro
Av. Pasteur 250, Rio de Janeiro, RJ, 22290-240,Brazil
lucena@openlink.com.br

Nelson Maculan
Programa de Engenharia de Sistemas e Computação, COPPE
Universidade Federal do Rio de Janeiro
P.O. Box 68511, Rio de Janeiro, RJ, 21945-970, Brazil
maculan@cos.ufrj.br

November 6, 2000

Abstract

A Lagrangian relaxation based exact solution algorithm for the Vehicle Routing Problem is introduced in this paper. Lower bounds are obtained by allowing exponentially many inequalities as candidates to Lagrangian dualization. Three different families of strong valid inequalities (each one with exponentially many elements) appear in the formulation used. For each family, violated inequalities are identified through separation procedures for points that define minimum K-Trees (i.e. the solutions to the underlying Lagrangian problems). Violated inequalities are dualized in a relax and cut framework. Upper bounds for the problem are generated through a simple Lagrangian based Clarke and Wright heuristic. The lower and upper bounds thus obtained are used in some variable fixation tests based on (approximating) linear programming reduced costs. Computational results indicate that the algorithm is competitive with other exact solution algorithms in the literature.

Key words: Exact Solution Algorithm, Vehicle Routing Problem, Valid Inequalities, Relax and Cut.

1 Introduction

Vehicle Routing Problems is the generic name given to a large class of problems involving the distribution of goods, services, information or personnel. A particularly important special case of the general problem is that of minimizing the total distance (time) required by a fleet of *vehicles* to satisfy delivery orders placed by *customers*. Vehicles are stationed at a central depot (to which they should return at the end of the operation) and have identical, fixed, capacities. The number of vehicles to be used, i.e. *routes* to be formed, $K \geq 1$, is assumed to be part of the problem input data. *Routing* decisions involve the allocation of customers to vehicles (without exceeding vehicle capacities) and determining the sequence in which customers allocated to a vehicle should be visited. This problem has been known in the literature as either the the Vehicle Routing Problem (VRP) (Christofides, Mingozzi and Toth [12, 13]), the Truck Dispatching Problem (Dantzig and Ramser [17], Christofides and Eilon [11], Krolak, Felts and Nelson [31])), the Vehicle Scheduling Problem (Clarke and Wright [14], Gaskell [24]), or else the Classical VRP (Laporte [33]). In this paper, the problem is to be referred, simply, as the VRP.

The VRP is clearly $\mathcal{NP}$-hard. Indeed it generalizes the Traveling Salesman Problem (TSP) (see the book by Lawler, Lenstra, Rinnooy Kan and Shmoys [38] or the survey by Jünger, Reinelt and Rinaldi [28], for instance) in that a TSP can be viewed as a VRP with a single vehicle of infinite capacity. The VRP has been the object of a considerable amount of research for the past four decades. For surveys on exact and approximate VRP algorithms the reader is referred to Christofides [10], Laporte and Norbert [35], Fisher [23] and Laporte and Osman [37]. An annotated bibliography on the subject can be found in Laporte [33]. VRP models and exact solution algorithm based on these models are studied in Beasley, Lucena and Poggi de Aragão [8]. Some recent exact solution algorithms for the problem can be found in Augerat, Belenguer, Benevant, Corberán, Naddef, and Rinaldi [3, 4], Hadjiconstantinou, Christofides and Mingozzi [29]), Mingozzi, Christofides and Hadjiconstantinou [45], Miller [44] and Ralphs, Pulleyblank and Trotter [48].

In this paper, one builds upon the results in [21], where a Lagrangian relaxation based exact solution algorithm for the VRP is proposed. Lower bounds in [21] are obtained from K-Tree relaxations of the problem. A K-Tree (see [22] for details) can be thought of as a generalization of the concept of a 1-Tree (see Held and Karp [26]). In this paper, the basic Lagrangian relaxation algorithm of Fisher [21] is modified in a number of ways. Firstly the algorithm proposed here, contrary to that of Fisher, is a *relax and cut* one (see Lucena [41, 42]). Therefore, an exponential number of inequalities are true candidates for Lagrangian dualization. Secondly, a stronger formulation is used. This formulation incorporates additional

families of valid inequalities to the formulation in [21]. Thirdly, tests for fixing variables, based on lower bounds for Linear Programming *reduced costs* are proposed and successfully used. In a further development, a quite effective Lagrangian heuristic, based on the algorithm of Clarke and Wright [14], is also introduced in this paper. The combination of all these ingredients leads to sharper lower bounds than those obtained in [21]. A more detailed description of most of what is presented here can be found in Martinhon [43].

This paper is organized as follows. In Section 2 some basic results from [21, 22] are presented. These are fundamental for the development of the algorithm introduced here. In particular, a VRP formulation is presented and its associated K-Tree relaxation is described. Section 3 brings a general outline for a relax and cut algorithm. For Sections 4, 5 and 6, given a point x in a multidimensional real valued space, this is assumed to be the incidence vector of a K-tree. In Section 4 a procedure for the separation of generalized subtour elimination constraints from x is proposed. In Section 5 the same applies in relation with comb inequalities. In Section 6, yet again, the same applies in relation with multistar inequalities. Variable fixation is the subject of Section 7. In Section 8 a Lagrangian based heuristic for the problem is introduced. In Section 9 one finds a description of the way implicit enumeration is conducted. Computational experiments are treated in Section 10. Finally Section 11 closes the paper with some conclusions that can be drawn.

2 VRP Formulation and Minimum K-Tree Relaxation

Let $G = (V_0, E)$ denote an undirected graph with a vertex set V_0 with $n+1$ vertices indexed $\{0, 1, \ldots, n\}$ and an edge set E. For simplicity, assume G to be a complete graph. Furthermore assume $V \subset V_0$ to be that subset of vertices indexed by $\{1, \ldots, n\}$. Following [21] vertices in V are associated with customers and the vertex indexed by 0 is associated with the depot. Edges in E may be referred to as either $e \in E$ or else $e = (i, j),\ i < j$, for $i, j \in V_0$; whatever is more convenient. Costs $\{c_e : \ e \in E\}$ which may express time, distance or a combination of both, are associated with the edges of G. For easy of presentation one will be somewhat lax with the notation and use i to denote, simultaneously, vertex $i \in V_0$ and that vertex in V_0 indexed by i, assuming, as a result, that those two vertices are the same. The same convention will also apply to an edge $e \in E$ and that edge of E indexed by e. Therefore, the demand for customer (vertex) $i \in V$ is given by d_i and the capacity for each one of the K vehicles stationed at the depot is given by b. The depot is assumed to have a demand of $d_0 = 0$ units. For a given set of vertices $S \subseteq V$, $d(S)$ denotes the sum of the demands for

the vertices in S. Accordingly, $E(S) \subseteq E, S \subset V_0$, is used for the set of edges of G with both endpoints in S. The complement in V_0 for a set of vertices $S \subseteq V$ is denoted $\overline{S} = V_0 \setminus S$. Those edges with one endpoint in $S \subseteq V$ and the other endpoint in $\overline{S}$ form the cut-set $E(S, \overline{S})$. Given a set of vertices $S \subseteq V$, the subset of edges of E with exactly one endpoint in S is denoted by $\delta(S)$. For simplicity, whenever set S has a single vertex, say vertex i, one uses $\delta(i)$ instead of $\delta(\{i\})$. Finally, $r(S)$ is used to denote $\lceil d(S)/b \rceil, S \subseteq V$, where $\lceil y \rceil$ is the smallest integer larger than or equal to the scalar y.

A K-tree, T_K, as introduced in Fisher [22] is defined as a subgraph of G with $n + K$ edges that spans the vertices in V_0. VRP lower bounds have been generated in [21] through the use of minimum K-trees with $2K$ edges incident on the depot vertex. Given G, an $O(n^3)$ algorithm for generating a minimum K-Tree can be found in Fisher [22].

In order to introduce a formulation for the VRP, let binary $\{0, 1\}$ variables $\{x_e : \ e = (i, j) \in E\}$ control the inclusion or not of an edge in a VRP solution ($x_e = 1$ for inclusion). Therefore, in accordance with the convention set above, those variables may also be described as $\{x_{ij} : i = 0, \ldots, n-1; j = (i+1), \ldots, n; i < j\}$. A set X is used to denote all the incidence vectors of K-trees with exactly $2K$ edges incident on vertex 0. A formulation for the VRP [21] (which precludes the use of single routes, i.e. routes serving a single customer) is given by

$$\min \left\{ \sum_{e \in E} c_e x_e : x \in \mathcal{R}_0 \right\}, \tag{1}$$

where $\mathcal{R}_0$ is the feasibility region defined by

$$\sum_{e \in \delta(i)} x_e = 2, \qquad i \in V, \tag{2}$$

$$\sum_{e \in E(S, \overline{S})} x_e \geq 2r(S), \qquad S \subseteq V, \tag{3}$$

$$x \in X. \tag{4}$$

Inequalities (3) are known as the Generalized Subtour Elimination Constraints (GSECs) and have been introduced by Laporte and Norbert [34] in the following equivalent description:

$$\sum_{e \in E(S)} x_e \leq |S| - 2r(S), \qquad S \subseteq V. \tag{5}$$

Inequalities (5) generalize the Subtour Elimination Constraints, introduced by Dantzig, Fulkerson and Johnson [16] for the TSP. Notice that $r(S)$

gives a lower bound on the number of vehicles necessary to feasibly service the customers represented in S. Inequalities (3) can be made stronger by replacing $r(S)$ with the optimal solution to the bin packing problem associated with bins of size b and a set of $|S|$ elements with weights $\{d_i : \ i \in S\}$. This strengthening of (3) has been credited by Laporte and Norbert [35] to Bezalel Gavish. In order to introduce another possible strengthening of (3) consider an auxiliary set $S^{'} = \{j \in \overline{S} \setminus \{0\} : d_j > br(S) - d(S)\}$, in relation with a given vertex set $S, |S| \geq 2$. Inequalities in (3) may be lifted (as suggested in [21]) to

$$\sum_{e \in E(S,\overline{S})} l_e x_e \geq 2r(S), \qquad S \subseteq N, \ \ |S| \geq 2, \tag{6}$$

where $\{l_e = 1 : \ \ e \in E(S,\overline{S}) \setminus E(S,S^{'})\}$ with the remaining coefficients (for the edges in $E(S,S^{'})$) being set either to 0 if $|S^{'}| \leq 2$ or else to $r(S)/(r(S)+1)$.

Given a feasible VRP route, an incidence vector x associated with this route has an entry $x_e = 1$ for an edge e in the route and $x_e = 0$, otherwise. The lifted Inequalities in (6) have been shown by Fisher [21] not to be facet inducing for the VRP polytope, i.e. the polytope defined by the convex hull of the incidence vectors associated with feasible VRP solutions. In what follows, GSECs are to be used in their lifted form (6).

2.1 K-Tree Relaxation

Suppose that one attaches Lagrangian multipliers $u_i \in I\!R$ to (2) and $v_S \in I\!R_+$ to (6). Then, after dualizing (2) and (6) in a Lagrangian fashion, one is left with the problem

$$z_D(u,v) = \min \left\{ \sum_{e \in E} \overline{c}_e x_e : x \in X \right\} \tag{7}$$

where, for $e = (i,j)$, $\overline{c}_e = c_e - u_i - u_j - \sum_{\left\{S \subseteq V: \ \ e \in E(S,\overline{S})\right\}} l_e v_S$.

At least in principle, the Subgradient Method (SM) of Held, Wolfe and Crowder [27] could be used to attain

$$\max_{u \in I\!R^n, v \in I\!R_+^{2^n-(n+1)}} z_D(u,v). \tag{8}$$

Nevertheless that should prove, practically speaking, unlikely. Notice that the number of entries in v would clearly grow exponentially with n. Therefore just the task of computing all subgradients

$$\sum_{e \in E(S,\overline{S})} 2r(S) - l_e x_e, \qquad S \subseteq V, \ \ |S| \geq 2, \tag{9}$$

for just a single iteration of the SM, may result, computationally speaking, unattainable. Even assuming that this first hurdle could be passed, obstacles would still remain. For instance, notice that, for any nontrivial VRP instance, the number of nonzero subgradients involved, at any iteration, is likely to be huge. As a result, multiplier values would be likely to change only very marginally from iteration to iteration and convergence to (8) could be jeopardized.

Fisher [21] dealt with the difficulties outlined above by selecting, *a priori*, a sufficiently small, attractive, subset of all GSECs to work with. The selection process is initialized with a feasible VRP solution (see Laporte and Osman [37] for a choice of VRP heurirstics). One would then choose $m = K + 3$ vertices in V_0 to act as *seeds*. The first K seeds are chosen as the vertices which are the furtherest away from the depot for each one of the K routes in hand. The three remaining seeds are chosen, sequentially, as those vertices that are maximally distant, simultaneously, from existing seeds and the depot (see [21] for details). All GSECs that are to be considered must originate from one of these $K + 3$ seeds. For each seed, up to 60 subsets $S \subseteq V$ are generated. In more detail, let s_i be a given seed, defined by vertex $i \in N$. Additional, let $i_1, i_2, \ldots, i_{n-1}$ be an indexing of the remaining $n - 1$ vertices in V. Subsets are generated, nested around s_i, in a pattern of increasing subset cardinality, as $\{s_i, i_1\}, \{s_i, i_2\}, \{s_i, i_1, i_2\}, \{s_i, i_1, i_2, i_3\}, \{s_i, i_1, i_2, i_4\}, \{s_i, i_1, i_2, i_3, i_4\}, \ldots$, until a subset cardinality of 60 is reached.

An alternative way for conducting Subgradient Optimization, when faced with exponentially many inequalities that are candidates for Lagrangian relaxation, is explained next.

3 Relax and Cut

A large number of combinatorial optimization problems can be generically described as the linear integer program

$$\min\{cy : \ \ Ay \geq b, \ \ y \in Y\}. \tag{10}$$

Variables y in (10) are assumed to be binary $0 - 1$, i.e. $y \in \mathcal{B}^q, q \geq 1$. Furthermore, let $c \in \mathbb{R}^p, p \geq 1$, $b \in \mathbb{R}^q$, and A be a real valued matrix of conformable dimensions. Assume, as it is customary in Lagrangian relaxation, that

$$\min\{cy : \ \ y \in Y\} \tag{11}$$

is an easy (polynomial time) problem to solve. On the other hand, in what is unusual for the application of Lagrangian relaxation, let q be very large (even exponential in p). In spite of that, assume that one wishes to dualize

$$\{a_i y_i \geq b_i : \ \ i = 1, 2, \ldots, q\} \tag{12}$$

in a Lagrangian fashion and let $\Lambda \in \mathbb{R}^q_+$ be the corresponding array of Lagrangian multipliers. Subgradient Optimization (SO) could then be used to solve

$$\max_{\Lambda \geq 0} \Big\{ \min\{(c - \Lambda A)y + \Lambda b : \quad y \in Y\} \Big\}. \tag{13}$$

Optimization is typically conducted here in an interactive way with multipliers being updated so that the optimal value of (13) is attained (see [27]). For the sake of completeness, let us briefly review the Subgradient Method (SM) (see [27] for details) which is used here. At any given iteration of the SM, for given feasible values of Lagrangian multipliers Λ, let $\overline{y}$ be an optimal solution to

$$\min\{(c - \Lambda A)y + \Lambda b : \quad y \in Y\} \tag{14}$$

Denote by z_{lb} the value of this solution and let z_{ub} be a known upper bound on (10). Additionally, let $G \in \mathbb{R}^q$ be an array of the subgradients associated with the relaxed constraints. For the current solution, $\overline{y}$, G is evaluated as

$$g_i = (b_i - a_i \overline{y}), \quad i = 1, 2, \ldots, q. \tag{15}$$

In the literature (see [20], for instance) Lagrangian multipliers are usually updated by firstly determining a "step size" θ,

$$\theta = \frac{\pi(z_{ub} - z_{lb})}{\sum\limits_{i=1,\ldots,q} g_i^2}, \tag{16}$$

where π is a real number assuming values in $(0, 2]$. One would then proceed to computing

$$\lambda_i \equiv \max\{0; \lambda_i + \theta g_i\}, \quad i = 1, \ldots, q \tag{17}$$

and then move on to the next iteration of the SM.

Under the conditions imposed here, the straightforward use of updating formulas (15)-(17) is not as simple as it might appear. The reason being the very large number of inequalities that would, typically, be dualized. In actual fact, for the conditions imposed here, at every SM iteration, subgradients could be divided into three groups. The first one would involve subgradients for those inequalities in (12) that are violated by $\overline{y}$. The second group would involve those subgradients g_i for which, currently, $\lambda_i > 0$. Notice that a subgradient may be, simultaneously, in the two groups just described. Finally, the third group consists of the remaining subgradients and their evaluation would account for the lion's share of the computational

burden at a SM iteration. One should notice that, under the proposed classification, subgradients may change groups from one SM iteration to another. It should also be noticed that the only multipliers that may directly contribute to the Lagrangian costs $(c - \Lambda A)$, at a given SM iteration, are the ones associated with subgradients in groups one or two. These multipliers are denoted *active multipliers* and their associated subgradients *active subgradients*. Conversely, subgradients on the third group are denoted *inactive* and so are their associated multipliers. Finally, another point to be made is that, from (17), no multiplier associated with subgradients in group three will change its current null value, after the application of that updating formula.

If, on the one hand, inactive multipliers do not contribute to Lagrangian costs, inactive subgradients, on the other hand, do play a decisive role in determining the value of θ. Typically, for our application, θ would tend to be extremely small, leaving multiplier values virtually unchanged from iteration to iteration. Bearing this in mind, one may choose to apply (15)-(17) exclusively to active subgradients and multipliers. That results into a dynamic scheme where the set of active multipliers may continuously change. Notice, in association, that a multiplier may become active at one given SM iteration, then become inactive at a subsequent SM iteration and, yet again, become active at a later iteration. The introduction of this scheme (very much akin to cutting planes generation) into implicit enumeration for a problem with exponentially many inequalities as candidates for Lagrangian dualization, has been firstly proposed and successively applied to the Steiner Problem in Graphs by Lucena ([41, 42]). Later on this approach has been used by Hunting, Faigle and Kern [30] for the Edge-Weighted Clique Problem and by Moraes Palmeira, Lucena and Porto [46] for the Quadratic Knapsack Problem.

A different approach, which has been introduced after [41, 42], and that also allows exponentially many inequalities as candidates for Lagrangian dualization, has been already coined *relax and cut* (see Escudero, Guignard and Malik [19] for details). That approach presents significant differences from the one in [41, 42]. Nevertheless, both approaches operate within a Lagrangian relaxation framework and dualize inequalities *on the fly*. Therefore the term *relax and cut* appears appropriate enough to describe the algorithm in [41, 42] as well and that is followed here.

For Lagrangian Relaxation based methods, the first attempt to allow exponentially many inequalities as candidates for dualization was suggested by Balas and Christofides [5]. For this approach, firstly, a valid relaxation of the problem being tackled is solved to optimality. Let $\overline{x}$ denote the relaxation solution. Then, a valid inequality which is violated at $\overline{x}$ is identified and dualized in a Lagrangian fashion. In the process, the Lagrangian multiplier associated with the violated inequality (initially valued at zero) is increased, in order to improve dual bound value, while keeping $\overline{x}$ optimal

for the modified Lagrangian problem. The procedure is repeated until no more violated inequalities which can improve dual bounds (while keeping $\overline{x}$ optimal for the modified Lagrangian problem) can be found. This overall scheme has been named *Restricted Lagrangian Approach*.

A second attempt to allow exponentially many inequalities as candidates for Lagrangian dualization was suggested by Gavish [25] for the Capacitated Minimum Spanning Tree Problem. The procedure begins by solving a given Lagrangian relaxation of the problem under consideration. The solution is examined in order to identify some valid inequalities that are violated. Violated inequalities are then added to the Lagrangian problem, resulting in a reinforced relaxation of the original problem. A dual ascent procedure is then applied and is complemented with subgradient optimization.

Finally, more recently, Fisher [21] proposed the dualization of an *a priori* selected subset of a family of valid inequalities (with exponentially many members). The approach suggested by Fisher goes halfway between the traditional Lagrangian relaxation framework and the one suggested in Lucena [41, 42].

As it may be appreciated, the approaches of Balas and Christofides [5] and Gavish [25] differ substantially from that of Lucena [41, 42], which bears a close resemblance with branch and cut type algorithms (see Padberg and Rinaldi [47]).

For the three following sections, separation algorithms for different families of VRP valid inequalities are described. One should stress that, in the presence of exponentially many inequalities as candidates to Lagrangian dualization, solving a separation problem is required in order to determine some of the active multipliers.

4 Separation of Generalized Subtour Elimination Constraints

Consider the following problem: given an optimal solution $\{\overline{x}_e : \ e \in E\}$ to the problem of finding a least cost K-Tree with an edge degree of $2K$ at vertex 0, find a GSEC that is violated at that solution or determine that no such inequality exists. This problem is clearly important to us since it identifies some of the GSECs to be labelled active. For obvious reasons, it is denoted *GSEC Separation Problem* (GSEC-SP). Fisher [21] dealt with a simplified version of this problem since the subsets of GSECs that are candidates for dualization in [21] comprise only a tiny fraction of all GSECs. As a result, the associated, restricted, GSEC-SP in [21] could be solved by inspection. For the general case, nevertheless, inspection is clearly not an option.

In this paper a two steps exact solution algorithm for the GSEC-SP is introduced. Let T_K be the *support graph* associated with $\overline{x}$, i.e. the subgraph

of G for those edges $e \in E$ with $\overline{x}_e = 1$. Therefore, T_K defines a K-Tree with $2K$ edges incident on vertex 0 (i.e. the depot). In what follows, given a subset $F \subseteq E$, one uses $x(F)$ to denote $\sum_{e \in F} x_e$.

In the first step, remove from T_K all edges that are incident on vertex 0. Let m be the number of connected components that result from this action and let $M = \{1, \ldots, m\}$ be an associated index set. Assume, without loss of generality, that one is working with (3) instead of (6). This is clearly valid since violation of (3) implies violation of (6). Let $S_k \subseteq V$, $k \in M$, denote the set of vertices in that component indexed by k. In association, compute $\sigma_k, k \in M$, where

$$\sigma_k = 2r(S_k) - \overline{x}(\delta(S_k)), \tag{18}$$

and, as defined before, $r(S_k) = \lceil d(S_k)/b \rceil$. Notice that whenever σ_k is strictly positive, the corresponding GSEC in (3) is violated for $\{\overline{x}_e : \ e \in E\}$ and should therefore be labelled active and dualized (in case the inequality has not been labelled active already).

In the second step, one selects all those K-Tree vertices with an edge cardinality of one. It is straightforward to verify that each of them define a violated GSEC which should be dualized.

The overall identification procedure described above could be summarized into the following algorithm:

Algorithm SSP (Subtour Separation Procedure):
Begin

1. Generate connected components from T_K, identify $\{S_k : \ k \in M\}$ and compute $\{\sigma_k : \ k \in M\}$. If $\sigma_k > 0, k \in M$, the GSEC implied by S_k is violated for $\{\overline{x}_e : \ e \in E\}$ and should be labelled active.

2. Identify K-Tree vertices of cardinality one. Label the GSECs implied by each of these vertices active.

End

After applying algorithm SSP, one should dualize active inequalities, in case they have not been dualized already.

In order to prove that algorithm SSP solves GSEC-SP exactly, let $E_T(S) \subseteq E$ define those edges with $\overline{x}_e = 1, e \in E(S)$. Accordingly let $V_k = S_k \cup \{0\}, k \in M$. Once again, in order to simplify the presentation, and yet again without loss of generality, let us consider inequalities (3) instead of (6).

Lemma 1 *If no S_k with $\sigma_k > 0$ exists for $k \in M$ then $d(S_k) \leq b$ for any $k \in M$.*

Proof: Suppose there exists an $S_l, l \in M$, with $\overline{x}(\delta(S_l)) = 1$. Since $\lceil d(S_k)/b \rceil \geq 1$ for any $k \in M$, one would then have $\sigma_l = 2\lceil d(S_l)/b \rceil - 1 > 0$. That, in turn, contradicts the assumption that $\sigma_k \leq 0, \forall k \in M$. Therefore,

$$\overline{x}(\delta(S_k)) \geq 2 \text{ for all } k \in M. \tag{19}$$

Suppose now, by contradiction, that $d(S_l) > b$ for some $l \in M$. Then $\lceil d(S_l)/b \rceil \geq 2$ and, since no violated GSEC is meant to exist, $\sigma_l = 2\lceil d(S_l)/b \rceil - \overline{x}(\delta(S_l)) \leq 0$. That, in turn, translates into

$$\overline{x}(\delta(S_l)) \geq 4. \tag{20}$$

At this point, notice that any connected component $E_T(S_k), k \in M$, must have $|E_T(S_k)| \geq |S_k| - 1$. Therefore, from (19) and (20), $|E_T(V_k)| \geq |S_k| + 1$, $k \in M \setminus \{l\}$ and $|E_T(V_l)| \geq |S_l| + 3$, thus resulting in $\sum_{k \in M} |E_T(V_k)| \geq \sum_{k \in M \setminus l}(|S_k| + 1) + |S_l| + 3 = (m + 2) + n$. The minimum possible number of edges induced by solution $\overline{x}$ would then sum, exactly, $m + n + 2$ and that is attained for

$$\overline{x}(\delta(S_k)) = 2, \quad k \in M \setminus \{l\}, \tag{21}$$

and

$$\overline{x}(\delta(S_l)) = 4. \tag{22}$$

One would thus have component l being linked to vertex 0 by exactly 4 edges and each the remaining components being linked to vertex 0 by exactly 2 edges. Furthermore, since the topology implied by T_K is that of a K-Tree one must additionally have $m = K - 1$. As a result, $\sum_{k \in M} |E_T(V_k)| \geq n + K + 1$, in contradiction with the fact that a K-Tree must have, exactly, $n + K$ edges. Therefore, $d(S_k) \leq b$, for all $k \in M$. □

Lemma 2 *If no violated component exists at the end of algorithm SSP, then every connected component $S_k, k \in M$, must be linked to vertex 0 by exactly 2 edges. Furthermore, every $S_k, k \in M$, must define a tree.*

Proof: If no violated component exists at the end of algorithm IDS then, from Lemma 1, $d(S_k) \leq b$ for all $k \in M$. It then follows that $\sigma_k = 2 - \overline{x}(\delta(S_k)) \leq 0$ for all $k \in M$ and, consequently, $\overline{x}(\delta(S_k)) \geq 2$ for all $k \in M$.

Suppose, by contradiction, that $\overline{x}(\delta(S_l)) \geq 3$ holds for some $l \in M$ and notice that, due to connectivity, $\overline{x}(E(S_k)) \geq |S_k| - 1$ for all $k \in M$. It then follows that $|E_T(V_k)| \geq |S_k| + 1$ for $k \in M \setminus \{l\}$, while $|E_T(V_l)| \geq |S_l| + 2$. On the other hand, one has exactly m connected components and therefore $\sum_{k \in M} |E_T(V_k)| \geq \sum_{k \in M \setminus \{l\}}(|S_k| + 1) + |S_l| + 2$ must hold. Nevertheless, as $\sum_{k \in M} |S_k| = n$ and $\sum_{k \in M} |E_T(V_k)| = n + K$ it follows that $n + K \geq n + m + 1$ and $m \leq K - 1$ is then implied. Recalling that the number of

connected components equals m, denote by ψ the number of edges that are non incident on the depot vertex. A valid lower bound on ψ is therefore $\sum_{i \in M}(|S_i| - 1) = n - m$. Nevertheless, since $m \leq K - 1$, it then follows that $\psi \geq n - K + 1$. That contradicts the fact that one has a K-Tree with exactly $n - K$ edges non incident on the depot.

An analogous proof can be followed to show that any of the connected components implied by $S_k, k \in M$, must have exactly $|S_k| - 1$ edges when no violated components are found at the end of SSP. In that line, suppose, in order to reach a contradiction, that one of the connected components, say that one implied by S_l, has at least $|S_l|$ edges. Notice that every connected component $S_k, k \in M$, must have, by connectivity, at least $|S_k| - 1$ edges. Since $\overline{x}(\delta(S_k)) = 2$ for all $k \in M$ and additionally $m = K$, one reaches the strict inequality $\sum_{k \in M} |E_T(V_k)| = \sum_{k \in M}(\overline{x}(E_T(S_k)) + 2) > \sum_{k \in M}(|S_k| - 1 + 2) = \sum_{k \in M} |S_k| + m = n + K$. That, in turn, contradicts the fact that $\sum_{k \in M} |E_T(V_k)| = n + K$ and therefore every connected component implied by $S_k, k \in M$, must define a tree. $\square$

Theorem 1 *Let T_K be the support graph associated with $\overline{x}$, i.e. a K-Tree with $2K$ edges incident on vertex 0. Algorithm SSP will always find a violated subtour in T_K, provided one exists.*

Proof: Suppose that no violated partition $S_k, k \in M$, has been found at the end of step 1 of algorithm SSP. Therefore, in order to conclude that no violated GSEC exists on T_K, one must ensure that no vertex with an edge degree of one exists on that tree (step 2 of algorithm SSP).

Provided no violated GSEC has been found in step 2 of algorithm SSP, all T_K vertices must have an edge degree of, at least, 2. That follows since a T_K vertex with an edge degree of one would have been detected in step 2 of SSP. Furthermore, lemma 2 establishes that one must have exactly two edges on T_K connecting component S_k to the depot vertex. Denote by $(0, v_1)$ and $(0, v_2)$ the corresponding two edges and $G_k = (V_k, E_T(V_k))$ the implied subgraph. Since S_k defines a tree (lemma 2), the existence of one only cycle in G_k is implied. Denote by C_k the vertices on this cycle. Being a tree, $E_T(S_k)$ must contain one only path between vertices v_1 and v_2. Assume there exists a vertex with an edge degree of at least 3 in such a path. It is then easy to check that $\overline{V_k} = V_k \setminus C_k$ is non empty. Therefore $E_T(\overline{V_k})$ would define a forest. That in turn would imply the existence of a vertex with an edge degree of one in $E_T(V_k)$, thus contradicting previous assumptions. Since capacity constraints are satisfied for every S_k and $E_T(C_k) = E_T(V_k)$ for every $k \in M$ is implied, it then follows that T_K must define a feasible VRP solution. $\square$

Additional violated GSECs can be obtained by further refining algorithm SSP. To that effect some definitions are required.

Definition 1 *A K-Tree leaf is an edge of a K-Tree with one end vertex with edge cardinality one. Given a K-Tree leaf, denote a leaf base the leaf vertex with edge cardinality larger than one.*

Additional violated GSECs can be obtained by shrinking K-Tree leaves into their leaf bases. That applies when one has exactly $l-1$ leaves incident on a leaf base with an edge degree of $l \geq 2$. The original vertices of G mapped into a shrunken vertex define a new violated GSEC. This overall scheme is to be repeated for as long as it is applicable.

5 Separation of Combs

A problem akin to the one just described for the separation of GSECs can also be posed in relation with Comb Inequalities. Differently from the exact solution approach used in the previous section, a heuristic will be suggested for the separation of comb inequalities. The comb inequalities studied here are those suggested in Cornuejols and Harche [15]. Namely, let H be a set of vertices denoted *handle* and $T_1, \ldots, T_s$ be a collection of sets of vertices denoted *teeth*. A particular type of comb inequality, which is valid for the VRP, is given by

$$x(E(H)) + \sum_{i=1}^{s} x(E(T_i)) \leq |H| + \sum_{i=1}^{s} |T_i| - \frac{3s+1}{2} + \alpha(K-1) \qquad (23)$$

where, for $s \geq 3$ and odd,

$$|T_i \setminus H| \geq 1 \text{ and } |T_i \cap H| \geq 1 \text{ for } i = 1, \ldots, s,$$

$$|T_i \cap T_j| = 0, 1 \leq i < j \leq s$$

and α may assume values 0, 1 or 2. A value $\alpha = 0$ applies when $0 \notin H \cup (\cup_{i=1}^{s} T_i)$. When $0 \in (H \setminus \cup_{i=1}^{s} T_i)$, $\alpha = 1$ applies. Finally, $\alpha = 2$ when $0 \in H \cap T_i$ for some $i = 1, \ldots s$.

Specific separation algorithms have been devised for combs where α equals either 0 or 1. The case where α equals 2 may obviously be considered as well. However, that case has been left out of this study since the trade off between the computational effort associated with their use the dual bound improvements they bring about does not appear to pay off.

Definition 2 *An external K-Tree leaf is a K-Tree leaf that is not incident on the depot vertex.*

Let T_K be a K-Tree with $2K$ edges incident on the depot. In association, let Q be a set of non intersecting external leaves of T_K. Furthermore, suppose that T_K is the support graph associated with $\{\overline{x}_e : e \in E\}$. A sufficient condition for the existence of a comb inequality which violates $\overline{x}$ is given bellow.

Theorem 2 *If $|Q| \geq 3$, then there exists a comb inequality which violates $\overline{x}$.*

Proof: Let $|Q| \geq 3$. Define H as being the set of all vertices in V_0 with a T_K edge degree greater or equal to 2. Additionally, select three different leaves of Q and denote them $T_i, i = 1, \ldots 3$. The proof consists in showing that a comb inequality, which violates $\overline{x}$, is implied by H, T_1, T_2 and T_3. To that order, let $\overline{Q}$, with cardinality η, be the set of all leaves in T_K (external or not). Clearly, $Q \subseteq \overline{Q}$. Recalling that T_K has exactly $n + K$ edges, one must have $\overline{x}(E(H)) = n + K - \eta$. Therefore, H must contain $n + K - \eta$ edges with both endpoints in H (the remaining η edges in T_K must belong to $\overline{Q}$) and $|H| = n + 1 - \eta$. Furthermore, one must have $\sum_{i=1}^{3} \overline{x}(E(T_i)) = 3$ and $\sum_{i=1}^{3} |T_i| = 6$. Since $\alpha = 1$, as $0 \in H$, it then follows that the difference between the left and the right hand sides of (23), say σ_C, must equal, exactly, 2. Therefore a comb inequality which violates $\overline{x}$ is thus implied by H, T_1, T_2 and T_3. $\square$

Assume that one is given a K-Tree T_K with $2K$ edges incident on the depot vertex. Furthermore, assume that T_K is the support graph for $\{\overline{x}_e : e \in E\}$. A heuristic that attempts to separate comb inequalities (in their simplest form, as described above) is proposed next.

Algorithm CSP (Comb Separation Procedure):
Begin

1. Introduce into H all those vertices with a T_K edge degree larger or equal to two.
2. Form a set Q with the non intersecting external leaves of T_K.
3. If $|Q| \geq 3$, choose 3 different, arbitrary, components of Q and denote them, respectively, T_1, T_2 and T_3. Sets H, T_1, T_2 and T_3 define a comb inequality that violates $\overline{x}$.

End

Given $\overline{x}$ and a comb inequality as in (23), let σ_C be the difference between the left and right hand sides of (23). Value σ_C can be interpreted as being the degree of violation for that inequality. Furthermore denote a *SmallComb* a comb inequality which has $\alpha = 0$ (i.e. it does not include the depot vertex). In our case, such an inequality can be thought of as being *local*, in the sense that it is associated with one only connected component obtained from T_K (after eliminating the edges incident on the depot vertex). Denote a *BigComb* a comb inequality with $\alpha = 1$ (i.e. it does not include the depot vertex). Such an inequality can be seen as being *global* in the sense that it involves more than one of the connected components referred above. In the same vein, one can define a *BigTooth* as being a tooth which consists, yet again, of a connected component (as referred above) with the additional property that it contains at least one cycle. Having such a kind of tooth, in a given comb inequality, appears desirable since it leads to a higher degree of violation (as it may be observed from (23)).

A number of alternatives to CSP are available in an attempt to identify comb inequalities which violate $\overline{x}$ for σ_C over 3. As one may appreciate from (23), for a given handle H, it is advantageous (in terms of increasing the degree of violation) to have as many odd K-Tree leaves into the inequality as possible.

Another alternative would be to construct BigCombs by firstly forcing BigTeeth to contain the largest possible number of cycles. As it may be appreciated from (23) that would increase the implied degree of violation. For a given connected component, consider the number of vertices and edges in it. Notice that the larger the ratio between the two, the larger is the number of cycles in the connected component.

For the computational results in section 10, all the comb separation procedures described above have been used.

6 Separation of multistars

Multistar inequalities, which induce facets of the VRP polytope when customer demands are identical, have been introduced by Araque (see Araque, Kudva, Morin and Pekny [2]). These inequalities remain valid for the problem under non identical customer demands. In this case, nevertheless, they are not any more facet inducing for the VRP polytope.

Consider a customer vertex $k \in V$ and let $\Gamma = \{S_i : i \in I_\Gamma\}, I_\Gamma = \{1, \ldots, s\}$, be a set of subsets of V. Elements of Γ must satisfy $d(S_i) \leq b$, for any $i \in I_\Gamma$, $S_i \cap S_j = \{k\}$ and $d(S_i \cup S_j) > b$ for any $i, j \in I_\Gamma, i \neq j$. Under these conditions,

$$\sum_{i \in I_\Gamma} x(\delta(S_i)) \geq 4s - 2 \tag{24}$$

is valid for the VRP and is denoted a multistar inequality (Γ, k). In relation with multistar inequality (24), vertex k is known as its *nucleus* while $S_i, i \in I_\Gamma$, are the multistar *extremities*.

Assume, as it was the case before, that the support graph of $\{\overline{x}_e : e \in E\}$ is a K-Tree $T_K = (V_0, E_T(V_0))$, with an edge degree of $2K$ at the depot vertex. In association, consider a tree $T = (V', E_T(V'))$ of T_K such that $\overline{x}(\delta(V')) = 1$. Denote by Φ the set of all such trees. A subset $\overline{\Phi} \subseteq \Phi$ of particular interest, is defined as $\overline{\Phi} = \{T = (V', E_T(V')) \in \Phi : d(V') \leq b\}$.

Definition 3 *A tree $T_B = (V_B, E_T(V_B))$ is denoted a basic extremity of $\overline{\Phi}$ if $d(V_B) \geq d(V'), \forall T = (V', E_T(V')) \in \overline{\Phi}$. Remaining trees are denoted non basic. For every set $\overline{\Phi}$, select only one basic extremity (in case of a draw, choose T_B arbitrarily).*

Definition 4 *A filter of $\overline{\Phi}$ is the set Φ_K with all those trees of $\overline{\Phi}$ that are not contained in the basic extremity.*

Since one aims at generating violated multistars, the underlying idea is to feasibly combine the basic extremity T_B with some other trees from Φ_K. A non basic extremity S_i of a multistar is generated by combining a nucleus k of V_B with a given tree $T = (V^{'}, E_T(V^{'}))$ of $\phi_K \setminus T_B$ so that $d(V^{'} \cup \{k\}) \leq b$. Obviously, in the process, care must be taken in the selection of a nucleus k (in order to enhance the chances of attaining multistar violation). A procedure for nucleus selection, which leads to multistar violation, is obtained from a result that follows.

Theorem 3 *Suppose that the extremities* $\Gamma = \{S_i : i \in I_\Gamma\}$ *of a violated multistar* (Γ, k) *have been generated using a filter* Φ_K *where* $s = |I_\Gamma| \geq 2$. *Let* $S_1 = V_B$ *and* $k \in V_B$ *be the nucleus of* (Γ, k). *Then, the edge degree* d_k *of nucleus* k *is such that* $d_k \leq 3$ *for any* $s \geq 2$.

Proof: In order to prove the theorem it suffices to notice that $\overline{x}(\delta(S_1)) = 1$ (since $S_1 = V_B$) and that $\overline{x}(\delta(S_i)) = d_k + 1$, for $i \in I_\Gamma \setminus \{1\}$. It then follows that

$$\overline{x}(\delta(S_1)) + \sum_{i \in I_\Gamma \setminus \{1\}} \overline{x}(\delta(S_i)) = 1 + (s-1)(d_k + 1). \tag{25}$$

Since constraint (25) is assumed to be associated with a violated multistar, one must have $1 + (s-1)(d_k + 1) < 4s - 2$ and therefore $d_k \leq (3s-2)/(s-1)$ applies. Furthermore, by induction, it is easy to establish that $3 < (3s-2)/(s-1) \leq 4$ for all $s \geq 2$. Since d_k must be integral and $d_k < (3s-2)/(s-1)$, one must have $d_k \leq 3, \forall s \geq 2$. □

From the results above, a procedure to separate multistars of 2 extremities is immediate. It should be noticed that separation of multistars with more than 2 extremities will incur into a considerable computational burden (due to the combinatorial nature of the underlying problem involved).

Algorithm S2M (Separation of 2 extremities multistars):
Begin

1. Generate filter Φ_K and set $S_1 = V_B$, as indicated above.
2. **For** (every vertex $k \in V_B$ with $d_k \leq 3$) **do**
   (a) **For** (every $T = (V^{'}, E_T(V^{'})) \in \Phi_K \setminus T_B$) **do**
      i. **If** $d(V_B) + d(V^{'}) > b$ and $d(V^{'}) + d(\{k\}) \leq b$ **then**
         A. The musltistar with extremities $S_1 = V_B$ and $S_2 = V^{'} \cup \{k\}$ is violated.
   (b) **End For**
3. **End For**

End

It is straightforward to adapt the above procedure for situations where more than two basic extremities are involved (thus increasing multistar violation). In this case, nevertheless, potentially, a very large number of violated multistars could be generated. Therefore, care must be exercised in order to achieve a reasonable trade off between computational effort and lower bound improvement. Bearing this in mind, in this study, one only takes into consideration violated multistars with 2 extremities and a nucleus k with a degree $d_k = 1$.

7 Variable Fixation

For the computational experiments carried out, tests for fixing variables have been implemented at every node of the search tree. The idea being to use VRP upper bounds and the K-Tree lower bounds to attempt to fix edges either in or out of a solution. In this way, roughly speaking, a minimum K-Tree (with an edge degree of $2K$ at the depot vertex) which is forced to contain a given edge is computed. Clearly, at the root node of the enumeration tree, if the cost of such a K-Tree exceeds a known VRP upper bound, the corresponding edge is guaranteed not to be part of an optimal VRP solution. The edge may then be eliminated from G. Conversely, a similar test may be implemented in order to check if a given edge must be part of an optimal VRP solution. In this way, a minimum K-Tree (with an edge degree of $2K$ at the depot vertex), which is forced not to contain a given edge, is computed. Clearly, at the root node of the enumeration tree, if the cost of such a K-Tree exceeds a known VRP upper bound, the corresponding edge is guaranteed to be part of an optimal VRP solution. The corresponding variable may then be fixed to one. Further details on variable fixation can be found in [43].

Fixing variables, as outlined above, would result, computationally speaking, far too expensive. Procedures that compute lower bounds on the Linear Programming *reduced costs* discussed above have, alternatively, been implemented.

8 Primal bounds: a Lagrangian based Clarke and Wright procedure

Throughout subgradient optimization, a Clarke and Wright [14] based procedure is used to generate feasible VRP solutions. The procedure is called a number of times; each one under a different set of (Lagrangian modified) edge costs. In this way one attempts to benefit, in the upper bounds generation, from the same source that brings about the relax and cut based lower

bounds introduced here.

The form in which this Clarke and Wright type algorithm is implemented presents some peculiarities. In addition to the use of Lagrangian modified costs (which could assume negative or positive values), one also makes room for the use of variable fixation information. A more detailed description of the algorithm is then necessary.

Let $\{c_{ij}^t : e = (i,j) \in E\}$ be the Lagrangian modified edge costs at iteration t of the Subgradient Method. Recall that $E(V) \subset E$ denotes all those edges of G that are not incident on the depot. In association, denote by $F^t(V)$ the subset of edges of $E(V)$ not yet fixed either to zero or one (i.e. the set of *free* edges at iteration t of SM). Let ξ be a VRP solution where vehicle capacity constraints are not violated but is infeasible since a number of routes larger than K are involved. Such a solution would typically emerge after a few edges have been fixed to one and either single vertex routes or else fixed edge routes are formed to initialize the Clarke and Wright algorithm. Define $E^t(\xi, V) \subset F^t(V)$ as being the set of edges for which the following conditions apply: (a) edge $(i,j) \in F^t(V)$ and i and j are on different routes of ξ; (b) the combined demand for the routes of ξ that contain vertices i and j must not exceed the vehicle capacity; (c) each one of the vertices i and j must be a *route extremity* (i.e. be linked to the depot through, at least, one edge in ξ). Savings $S_\xi^t = \{s_{ij}^t = c_{i0}^t + c_{j0}^t - c_{ij}^t : e = (i,j) \in E^t(V,\xi)\}$ could then be computed. Furthermore, Lagrangian modified costs for those edges incident on the depot, which have already been fixed to zero, are set to ∞. Notice that proceeding in this way one makes it impossible for a Lagrangian Clarke and Wright route to contain such an edge.

Differently from the traditional Clarke and Wright algorithm, one initializes the algorithm by introducing all the edges already fixed to one into the solution to be formed. Single routes formed by vertices which have already been proved not to be incident on the depot vertex are initially considered. Nevertheless, as the algorithm proceeds, they will be readily eliminated (due to their exceedingly high costs).

In order to simplify the description of the algorithm, define an initial *degree of freedom*, for every vertex $i \in V$, as being equal to two less the number of edges fixed to one which are incident on i. In association, notice that a vertex with an initial degree of freedom equal to zero can not be linked to other vertices while forming Clarke and Wright routes. Only vertices with degrees of freedom one or two can serve that purpose. Bearing this in mind, from this point on, the Lagrangian Clarke and Wright algorithm would proceed exactly as the classical one. At the point where a feasible VRP solution has been constructed, 3-opt moves (see Lin and Kerningham [39]) are applied for each individual route in the solution (in an attempt to reduce upper bound value).

A variant of the heursitic described above has also been implemented in this study. The basic difference amounting to the use of edge costs given

by $\{c_e(1-\overline{x}_e) : e \in E\}$ instead of Lagrangian ones. As a result, edges that appear at the dual solution are made more attractive to be chosen along the algorithm.

No significative difference has been noticed (in terms of bound quality) between the two versions of the heuristic. For some of the VRP instances tested, one would produce better bounds than the other while the reverse would be true for some other instances. The computational results quoted on Section 10 have been obtained by applying both heuristics, for every instance tested, at the root node of the enumeration tree. At nodes other than the root, only the version using Lagrangian costs has been used.

9 Branching rules

A branching rule, akin to the one used in Christofides, Mingozzi and Toth [12] and Miller [44] has been implemented in conjunction with the VRP lower and upper bounding schemes described above. The basic idea being to extended a partially formed feasible route initiated at the depot vertex. Therefore two *end points* exist for a route being formed, one of which is the depot vertex. It should be noticed that, due to the variable fixation tests, a partially formed route may, eventually, be extended by a chain of edges fixed to one (instead of a single edge).

Branching is carried out by firstly choosing an yet unscanned tree node with the least associated lower bound (ties are broken arbitrarily). At this point, one may be faced with some routes that have already been closed and other ones which are still under construction. A tree node is normally associated with a vertex from G which is an endpoint (other than the depot vertex) of a route being formed. An edge incident on this vertex is then chosen to extend the corresponding partially formed route. The edge to be chosen is the one with the least Lagrangian cost. Clearly, only the inclusion of those edges which will not overflow vehicle capacity is to be considered. Other types of infeasible (and more subtle) moves (particularly the ones which may bring, at a later stage, the kind of infeasibility quoted above) may also be detected when choosing the branching edge.

Variable fixation has been applied at search tree nodes other than the root. The information thus obtained is then carried over to lower level tree nodes.

10 Computational experiments

Computational experiments have been conducted in order to evaluate the quality of the lower and upper bounds proposed in this paper. Experiments were performed, under the LINUX operating system, on a Pentium III based machine with CPU running at 450 MHz and having 256 Mb of RAM memory.

The algorithms were coded in C and the -O3 option was used for compilation under GNU's gcc compiler.

A comparison with the lower bounds of Fisher [21] is detailed in Table 1. Columns on that table correspond, respectively, to instance descriptions (for instance, c51.dat, indicates a problem with 50 customers plus the depot), vehicle capacity, number of vehicles to be used, Fisher's lower bounds, three different relax and cut lower bounds (one involving GSECs, another one involving GSECs and combs and a third one involving GSECs, combs and mutistars), best Lagrangian Clarke and Wright bounds and best known upper bounds. An asterisk, for an entry in column 8 of Table 1, indicates that the corresponding upper bound has been proven to be optimal. VRP instances used in this experiment are the ones in [21]. Instances c51.dat, c76.dat and c101.dat come [11]. Instances c150.dat and c200.dat come from [21]. Instance c101b.dat come [12]. Instances c121.dat, f45.dat, f72.dat and f135.dat come from [21].

In order to allow for a proper comparison with the results quoted in [21], edge costs $\{c_e : e \in E\}$ were taken as the Euclidean distance between the corresponding pair of vertices (no rounding involved). Letter "a", for an entry in column 1, indicates that 3000 iterations of the SM were allowed for the relax and cut algorithms. For the remaining instances, only 2000 iterations were allowed. As customary, an initial value of 2.0 was associated with the step-size parameter α. After 50 consecutive SM iterations, without an overall improvement on the lower bounds, α is to be reduced to 0.75 of its current value. The Lagrangian Clarke and Wright heuristic has been called for every one of the initial 50being that it has been empirically observed that Lagrangian multipliers (and costs) tend to change only slightly after that many iterations. As a result, the feasible solutions thus obtained tend to remain virtually unchanged. Notice that this effect is less dramatic for the lower bounds since small perturbations for Lagrangian multiplier values tend to imply larger perturbations for lower bound values.

Table 2 gives the CPU times for computing the relax and cut bounds quoted in Table 1. A direct comparison with the CPU times quoted by Fisher [21] does not seem possible since quite different computers are used in each case.

As it can be appreciated from Table 1, substantial improvements over the lower bounds in [21] have been attained by each one of the three different relax and cut algorithms used. That appears to indicate that the relax and cut algorithms were capable of benefiting from the stronger VRP formulations associated with them. As to which relax and cut version performed best in terms of lower bound quality, the results appear inconclusive. If one brings into the picture CPU times, it does not seem to pay off, at least for the computational results obtained, to go for versions other than the one involving only GSECs and the degree constraints. For the computational results that follow, only that version has been used.

Further computational results are given in Table 3. They relate with the full blown branch and bound algorithm. The SM parameter settings, for the root node, are identical to those for Table 1 (2000 iterations case). For nodes other than the root, 1000 iterations of the SM are allowed with α being reduced, as above, after 50 consecutive iterations without an overall improvement of the lower bound. The Lagrangian Clarke and Wright heuristic, under Lagrangian modified costs, has been called at every tree node (for every one of the first 1000 iterations of the SM at the root node and for every one of the first 250 iterations at nodes other than the root). Instances A-n32-k5 and A-n46-k7 come from [3]. Instance att48.vrp come from [49]. Instance ch18.dat come from [15]. Instances eil7.vrp, eil21.vrp, eil23.vrp, eil30.vrp and eil33.vrp come from[11]. Instances mch16a, mch16b, mch21a, mch21b, mch22 and mch62 come from [45]. Instance mlm24 is a randomly generated instance introduced in this paper and instance rhg14.dat comes from [50].

All instances in Table 3 are generated in the Euclidean plane (with customers corresponding to points in that plane). Edge costs are taken as the associated Euclidean distances (with the addition of 0.5) rounded to the largest integer with a lesser value.

Based on the computational results obtained, the proposed algorithm appears competitive with the other exact solution VRP algorithms in the literature.

11 Conclusions

A relax and cut algorithm for the VRP was introduced in this paper. Three basic versions of the algorithm were computationally tested. Differences between the versions relate with the different families of strong valid inequalities that are used as candidate for Lagrangian dualization. For each one of these three families, separation procedures have been studied and implemented for points $\overline{x}$ with a support graph that defines K-Trees with degree $2K$ at the depot vertex. In particular, an exact solution algorithm is proposed for the separation of GSECs. Computational experiments indicate a substantial improvement over the lower bounds quoted by Fisher [21]. This comparison is important since one uses here the same K-Tree relaxation that has been introduced in [21]. A simple Lagrangian based Clarke and Wright heuristic and some variable fixation tests, based on (approximating) linear programming reduced costs, have also been proposed and computationally tested. Overall, the algorithm introduced in this paper appears competitive with the exact solution VRP algorithms in the literature.

References

[1] Y. Agarwal, K. Mathur and H. M. Salkin. A Set-Partitioning Based Exact Algorithm for the Vehicle Routing Problem. *Networks*, 19:731–749, 1989.

[2] 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.

[3] P. Augerat, J.M. Belenguer, E. Benevant, A. Corberán, D. Naddef, and G. Rinaldi. Computational results with a Branch and Cut Code for the Capacitated Vehicle Routing Problem. *Technical Report* RR940-M, Univ. Joseph Fourier, Grenoble, 1995.

[4] P. Augerat, J.M. Belenguer, E. Benevant, A. Corberán, D. Naddef, and G. Rinaldi. separating Capacity Constraints in the CVRP using Tabu Search. *European Journal of Operational Research*, 106:546–557, 1998.

[5] E. Balas and N. Christofides. A restricted Lagrangian approach to the traveling salesman problem. *Mathematical Programming*, 21:19–46, 1981.

[6] M. L. Balinski and R. E. Quandt. On an Integer Program for a Delivery Problem. *Operations Research*, 12:300–304, 1964.

[7] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh and P. H. Vance. Branch-and-Price: Column Generation for Huge Integer Programs. *Operations Research*, 46:316–329, 1998.

[8] J.E. Beasley, A. Lucena and M. Poggi de Aragão. The vehicle Routing Problem *Handbooks of Applied Optimization*, P. Pardalos and M.G.C. Resende eds., Oxford University Press, New York, to appear 2000.

[9] V. Campos, A. Corberan and E. Mota. Polyhedral Results for a Vehicle Routing Problem. *European Journal of Operational Research*, 52:75–85, 1991.

[10] N. Christofides. Vehicle Routing. In *The Traveling Salesman Problem*, E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan and D. B. Shmoys, eds., J. Wiley & Sons, Chichester, 1985.

[11] N. Christofides, S. Eilon. An Algorithm for the Vehicle-Dispatching Problem. *Operational Research Quarterly*, 20:309–318, 1969.

[12] N. Christofides, A. Mingozzi and P. Toth. Exact Algorithms for the Vehicle Routing Problem, Based on Spanning Tree and Shortest Path Relaxations. *Mathematical Programming*, 20:255–282, 1981.

[13] N. Christofides, A. Mingozzi and P. Toth. State Space Relaxation Procedures for the Computation of Bounds to Routing Problems. *Networks*, 11:145–164, 1981.

[14] G. Clarke and J. W. Wright. Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. *Operations Research*, 12:568–581, 1964.

[15] G. Cornuejols and F. Harche. Polyhedral Study of the Capacitated Vehicle Routing Problem. *Mathematical Programming*, 60:21–52, 1993.

[16] G. B. Dantzig, D. R. Fulkerson and S. M. Johnson. Solution of a Large Scale Traveling Salesman Problem. *Operations Research*, 2:393–410, 1954.

[17] G. B. Dantzig and J. H. Ramser. The Truck Dispatching Problem. *Management Science*, 6:80–91, 1959.

[18] J. Desrosiers, F. Soumis and M. Desrochers. Routing with Time Windows by Column Generation. *Networks*, 14:545–565, 1984.

[19] L. Escudero, M. Guignard, and K. Malik. A lagrangian relax and cut approach for the sequential ordering with precedence constraints. *Annals of Operations Research*, 50:219–237, 1994.

[20] M. L. Fisher. The lagrangian method for solving integer programming problems. *Management Science*, 27:1–18, 1981.

[21] M. L. Fisher. Optimal Solution of Vehicle Routing Problems using Minimum K-Trees. *Operations Research*, 42:626–642, 1994.

[22] M. L. Fisher. A Polynomial Algorithm for the Degree Constrained Minimum K-Tree Problem. *Operations Research*, 42:765–779, 1994.

[23] M. L. Fisher. Vehicle Routing Problem. In *Networks Models, Handbooks in Operations Research and Management Science*, T. L. Magnanti, C. L. Monma and G. L. Nemhauser, eds., Elsevier Publisher B.V., Amsterdam, 1995.

[24] T. J. Gaskell. Bases for Vehicle Fleet Scheduling. *Operational Research Quarterly*, 18:281–295, 1967.

[25] B. Gavish. Augmented Lagrangean based algorithms for centralized network design. *IEEE Transactions on Communications*, 33,12:1247-1257, 1985.

[26] M. Held and R.M. Karp. The traveling salesman problem and minimum spanning trees. *Operations Research*, 18:1138–1162, 1970.

[27] M. Held, P. Wolfe and H.P. Crowder. Validation Subgradient Optimization. *Mathematical Programming*, 6:62–88, 1974.

[28] M. Jünger, G. Reinelt and G. Rinaldi. The Traveling Salesman Problem. In *Networks Models, Handbooks in Operations Research and Management Science*, T. L. Magnanti, C. L. Monma and G. L. Nemhauser, eds., Elsevier Publisher B.V., Amsterdam, 1995.

[29] E. Hadjiconstantinou, N. Christofides, and A. Mingozzi. A new wxact solution algorithm for the vehicle routing problem based on q-paths and k-shortest paths relaxations. *Freight Transportation*, G. Laporte and M. Gendreau, eds., *Annals of Operations Research*, 61:21–44, 1995.

[30] M. Hunting, U. Faigle, and W. Kern. A Lagrangian relaxation approach to the edge-weighted clique problem. working paper, Department of Applied Mathematics, Twente University, 1998.

[31] P. D. Krolak, W. Felts and J. H. Nelson. A Man-Machine Approach Toward Solving the Generalized Truck-Dispatching Problem. *Transportation Science*, 6:149–170, 1972.

[32] A. Langevin, M. Desrochers, J. Desrosiers, S. Gélinas and F. Soumis. A Two-Commodity Flow Formulation for the Traveling Salesman and the Makespan Problems with Time Windows. *Networks*, 23:631–640, 1993.

[33] G. Laporte. The Vehicle Routing Problem. In *Annotated Bibliographies in Combinatorial Optimization*, M. Dell'Amico, F. Maffioli and S. Martello, eds., J. Wiley & Sons, Chichester, 1997.

[34] G. Laporte and Y. Norbert. A Branch and Bound Algorithm for the Capacitated Vehicle Routing Problem. *Operations Research Spektrum*, 5:77–85, 1983.

[35] G. Laporte and Y. Norbert. Exact Algorithms for the Vehicle Routing Problem. In *Surveys in Combinatorial Optimization, Annals of Discrete Mathematics, 31*, S. Martello, G. Laporte, M. Minoux and C. Ribeiro, eds., North-Holland, Amsterdam, 1987.

[36] G. Laporte, Y. Norbert and M. Desrochers. Optimal Routing Under Capacity and Distance Restrictions. *Operations Research*, 33:1050–1073, 1985.

[37] G. Laporte and I. H. Osman. Routing Problems: a Bibliography. In *Freight Transportation, Annals of Discrete Mathematics, 61*, G. Laporte and M. Gendreau, eds., North-Holland, Amsterdam, 1995.

[38] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan and D. B. Shmoys, eds. *The Traveling Salesman Problem*. J. Wiley & Sons, Chichester, 1985.

[39] S. Lin and B.W. Kerningham. An effective heuristic for the traveling salesman problem. *Operations Research*, 21:498–516, 1973.

[40] A. Lucena. *Exact Solution Approaches for the Vehicle Routing Problem*. Ph.D. Thesis, Imperial College, University of London, 1986.

[41] A. Lucena. Steiner Problem in Graphs: Lagrangean Relaxation and Cutting-Planes. *COAL Bulletin*, Mathematical Programming Society, 21:2–8, 1992.

[42] A. Lucena. Tight bounds for the Steiner problem in graphs. In *Proceedings of NETFLOW93*, pages 147–154, 1993.

[43] C. Martinhon. Lagrangian Relaxation with the Generation of Valid Inequalities Applied to the Vehicle Routing Problem, in Portuguese. *PhD Thesis*, COPPE, Federal University of Rio de Janeiro, 1998.

[44] D.L. Miller. A matching based exact algorithm for capacitated vehicle routing problems. *ORSA Journal on Computing*, 7:1–9, 1995.

[45] A. Mingozzi, N. Christofides, E. Hadjiconstantinou. An Exact Algorithm for the Vehicle Routing Problem Based on the Set Partitioning Formulation. Technical Report, Department of Mathematics, University of Bologna, 1994.

[46] M. de Moraes Palmeira, A. Lucena, and O. Porto. A relax and cut algorithm for quadratic knapsack problem. relatório técnico, Laboratório de Métodos Quantitativos, Departamento de Administração, Universidade Federal do Rio de Janeiro, 1999.

[47] M. Padberg and G. Rinaldi. A Branch-and-Cut Algorithm for the Resolution of Large-Scale Travelling Salesman Problems. *SIAM Review*, 33:60–100, 1991.

[48] T.K. Ralphs, W.R. Pulleyblank, and L.E. Trotter, Jr. On Capacitated Vehicle Routing. *CCOP Research Report*, TR 98-7, Cornell University, 1998.

[49] G. Reinelt. TSPLIB - A Traveling Salesman Problem Library. *ORSA Journal on Computing*, 3:376–384, 1991.

[50] D. Ryan, C. Hjorring and F. Glover. Extensions of the Petal Method for Vehicle Routeing. *Journal of the Operational Research Society*, vol. 44, 3:289–296, 1993.

[51] L. A. Wolsey. Column Generation Algorithms. In *Integer Programming*, chapter 11, John Wiley & Sons, Chichester, 1998.

Table 1: Lower bounds at the root node: Fisher's [21] instances

Problem	cap	nbv	Fisher	(S)	(SC)	(SCM)	LC&W	UB
c51.dat(a)	160	5	507.09	513.51	514.17	514.21	543.35	524.61*
c76.dat(a)	140	10	755.5	766.07	762.58	764.09	868.63	835.26
c101.dat(a)	200	8	785.86	792.47	788.75	791.84	850.02	826.14
c150.dat(a)	200	12	932.68	953.66	945.55	952.60	1083.61	1028.42
c200.dat(a)	200	16	1096.72	1150.23	1128.64	1138.81	1368.61	1291.45
c101b.dat(a)	200	10	817.77	817.56	817.57	816.68	819.56	819.56*
c121.dat	200	7	—	1003.86	1010.91	1007.25	1044.9	1042.11
f45.dat	2010	4	720.76	723.37	722.03	721.84	723.54	723.54*
f72.dat	30000	4	237.76	238.65	238.19	238.64	247.78	241.97*
f135.dat	2210	7	1133.73	1154.45	1147.39	1134.18	1174.85	1163.6

Table 2: CPU times (seconds) at the root node: Fisher's [21] instances

Problem	(S)	(SC)	(SCM)
c51.dat(a)	155	125	216
c76.dat(a)	363	400	633
c101.dat(a)	1040	1114	2816
c151.dat(a)	3934	4129	21449
c200.dat(a)	9592	9934	67009
c101b.dat(a)	418	546	2243
c121.dat	2719	3558	12161
f45.dat	117	107	206
f722.dat	750	803	856
f135.dat	4347	5189	15503

Table 3: Branch and Bound

Problem	cap	nbv	LB(root)	LC&W	CPU time (secs)	nb nodes	optimal
A-n32-k5	100	5	779.17	784	211	97	784
A-n46-k7	100	7	904.08	923	4612	626	914
att48.vrp	15	4	39075	40553	112192	4854	40002
c51.vrp	160	5	510.9	533	37676	3691	521
c101b.dat	200	10	819.01	822	14537	291	820
ch18.dat	10	2	741	741	4.4	1	741
eil7.vrp	3	2	114	114	0.01	1	114
eil21.vrp	6000	4	374.51	375	1.56	1	375
eil23.vrp	4500	3	569	569	16.3	1	569
eil30.vrp	4500	3	508.4	534	37864	10281	534
eil33.vrp	8000	4	828.4	837	479.3	114	835
f45.dat	2010	4	721.2	724	824.3	110	724
mch16a	55	5	322.06	333	34	86	333
mch16b	90	3	267.19	277	30	78	277
mch21a	58	6	426.01	430	11	16	430
mch21b	85	4	345.92	358	52	80	358
mch22	4000	6	479.08	495	430	422	495
mch26	48	8	606	606	4	1	606
mlm24	100	5	906.58	908	21	12	908
rhg14.dat	10	5	60.2	62	4	7	62