European Journal of Operational Research 59 (1992) 345-358
North-Holland

Invited Review

The Vehicle Routing Problem: An overview of exact and approximate algorithms

Gilbert Laporte

Centre de Recherche sur les Transports, Université de Montréal, C.P. 6128 Station A, Montréal, Canada H3C 3J7

Received May 1991

Abstract: In this paper, some of the main known results relative to the Vehicle Routing Problem are surveyed. The paper is organized as follows: (1) definition; (2) exact algorithms; (3) heuristic algorithms; (4) conclusion.

Keywords: Vehicle Routing Problem; survey

The *Vehicle Routing Problem* (VRP) can be described as the problem of designing optimal delivery or collection routes from one or several *depots* to a number of geographically scattered *cities* or *customers*, subject to side constraints. The VRP plays a central role in the fields of physical distribution and logistics. There exists a wide variety of VRPs and a broad literature on this class of problems (see, for example, the surveys of Bodin et al., 1983, Christofides, 1985a, Laporte and Nobert, 1987, Laporte, 1990, as well as the recent classification scheme proposed by Desrochers, Lenstra and Savelsbergh, 1990). The purpose of this paper is to survey the main exact and approximate algorithms developed for the VRP, at a level appropriate for a first graduate course in combinatorial optimization.

1. Definition

Let $G = (V, A)$ be a graph where $V = \{1, \dots, n\}$ is a set of vertices representing *cities* with the *depot* located at vertex 1, and A is the set of arcs. With every arc (i, j) $i \neq j$ is associated a non-negative *distance* matrix $C = (c_{ij})$. In some contexts, c_{ij} can be interpreted as a *travel cost* or as a *travel time*. When C is symmetrical, it is often convenient to replace A by a set E of undirected edges. In addition, assume there are m available vehicles based at the depot, where $m_L \leq m \leq m_U$. When $m_L = m_U$, m is said to be *fixed*. When $m_L = 1$ and $m_U = n - 1$, m is said to be *free*. When m is not fixed, it often makes sense to associate a fixed cost f on the use of a vehicle. For the sake of simplicity, we will ignore these costs and unless otherwise specified, we assume that all vehicles are identical and have the same capacity D. The VRP consists of designing a set of least-cost vehicle routes in such a way that

(i) each city in $V \setminus \{1\}$ is visited exactly once by exactly one vehicle;
(ii) all vehicle routes start and end at the depot;
(iii) some side constraints are satisfied.

The most common side conditions include:

(i) *capacity restrictions*: a non-negative weight (or demand) d_i is attached to each city $i > 1$ and the sum of weights of any vehicle route may not exceed the vehicle capacity. Capacity-constrained VRPs will be referred to as CVRPs;

(ii) the number of cities on any route is bounded above by q (this is a special case of (i) with $d_i = 1$ for all $i > 1$ and $D = q$);

(iii) *total time restrictions*: the length of any route may not exceed a prescribed bound L; this length is made up of intercity travel times c_{ij} and of stopping times δ_i at each city i on the route. Time- or distance-constrained VRPs will be referred to as DVRPs;

(iv) *time windows*: city i must be visited within the time interval $[a_i, b_i]$ and waiting is allowed at city i;

(v) *precedence relations* between pairs of cities: city i may have to be visited before city j.

This list is by no means exhaustive. A number of other interesting variants are described in the Golden and Assad (1988) book, for example. Here, we will mainly concentrate on CVRPs and on DVRPs.

2. Exact algorithms

Following the Laporte and Nobert (1987) survey, exact algorithms for the VRP can be classified into three broad categories: (i) direct tree search methods; (ii) dynamic programming, and (iii) integer linear programming. As the number of proposed algorithms is very large, we will provide six representative examples only: two direct tree search methods based on different relaxations, a dynamic programming formulation, and three integer linear programming algorithms. For some of the descriptions, we basically follow Laporte and Nobert (1987).

2.1. *The assignment lower bound and a related branch-and-bound algorithm*

The following algorithm due to Laporte, Mercure and Nobert (1986) exploits the relationship between the VRP and one of its relaxations, the m-TSP. Given the graph $G = (V, A)$ with a depot at vertex 1 and m vehicles based at the depot, the m-TSP consists of establishing m least-cost vehicle routes starting and ending at the depot, and in such a way that every remaining vertex is visited exactly once. As shown by Lenstra and Rinnooy Kan (1975), given an upper bound m_U on m, the m-TSP can be transformed into a 1-TSP as follows:

(i) Increase the number of vertices by introducing $m_U - 1$ artificial depots; let $n' = n + m_U - 1$, $V' = \{1, \dots, n'\}$ and $A' = A \cup \{(i, j): i, j \in V', i \neq j, i \text{ or } j \in V' \setminus V\}$.

(ii) The extended distance matrix $C' = (c'_{ij})$ associated with A' is defined by

$$c'_{ij} = \begin{cases} c_{ij} & (i,j \in V), \\ c_{i1} & (i \in V \setminus \{1\}, j \in V' \setminus V), \\ c_{1j} & (i \in V' \setminus V, j \in V \setminus \{1\}), \\ \gamma & (i,j \in (V' \setminus V) \cup \{1\}), \end{cases} \tag{1}$$

where the value of γ depends on the variant of the problem considered:
$\gamma = \infty$ yields the minimum distance for m_U vehicles;
$\gamma = 0$ yields the minimum distance for at most m_U vehicles;
$\gamma = -\infty$ yields the minimum distance for the minimum number of vehicles.

The VRP (CVRP, DVRP or both) can then be formulated as follows. Let x_{ij} $(i \neq j)$ be a binary variable equal to 1 if and only if arc (i, j) of A' appears in the optimal solution. If $d_i + d_j \geq D$, i and j

cannot belong to the same route and x_{ij} need not be defined.

$$\text{(VRP1)} \quad \text{minimize} \quad \sum_{i \neq j} c'_{ij} x_{ij} \tag{2}$$

$$\text{subject to} \quad \sum_{j=1}^{n'} x_{ij} = 1 \quad (i = 1, \ldots, n') \tag{3}$$

$$\sum_{i=1}^{n'} x_{ij} = 1 \quad (j = 1, \ldots, n'), \tag{4}$$

$$\sum_{i,j \in S} x_{ij} \leq |S| - v(S) \quad (S \subset V' \setminus \{1\};\ |S| \geq 2), \tag{5}$$

$$x_{ij} \in \{0, 1\} \quad (i,j = 1, \ldots, n';\ i \neq j). \tag{6}$$

In this formulation, (2), (3), (4) and (6) define a modified assignment problem (i.e. assignments on the main diagonal are prohibited). Constraints (5) are subtour elimination constraints: $v(S)$ is an appropriate lower bound on the number of vehicles required to visit all vertices of S in the optimal solution. These constraints are obtained by observing that for any $S \subset V' \setminus \{1\}$, $|S| \geq 2$, $\bar{S} = V' \setminus V$, we must have:

$$\sum_{i \in S} \sum_{j \in \bar{S}} x_{ij} \geq v(S),$$

and that the following identity holds:

$$|S| = \sum_{i,j \in S} x_{ij} + \sum_{i \in S} \sum_{j \in \bar{S}} x_{ij}.$$

The value of $v(S)$ depends on the type of VRP under consideration. In the CVRP, it is valid to take

$$v(S) = \left\lceil \frac{\sum_{i \in S} d_i}{D} \right\rceil.$$

In the DVRP, $v(S)$ is not as easy to determine *a priori*. Usually, however, a lower bound on its value can easily be determined during the course of a branch-and-bound process. Failing this, it is always valid to take $v(S) = 1$. It is worth observing that constraints (5) play a dual role: they ensure that all vehicle routes satisfy the capacity- or maximum-length restrictions; they also guarantee that the solution contains no subtour disconnected from the depot, since every subset S of $V \setminus \{1\}$ will be linked to its complement.

A natural algorithm now presents itself. Like the assignment-based algorithm for the TSP (see Laporte, 1992), the problem is solved through a branch-and-bound process in which subproblems are assignment problems. The only difference lies in the definition of illegal subtours. These now include

(i) subtours over a set S of vertices of $V \setminus \{1\}$;

(ii) vehicle routes violating capacity- or maximum-length restrictions: these consist of paths of vertices $(i_1, i_2, \ldots, i_r)$ where $i_1, i_r \in \{1, n+1, \ldots, n + m_{\text{U}} - 1\}$, $i_2, \ldots, i_{r-1} \in V \setminus \{1\}$ and $\sum_{t=2}^{r-1} d_{i_t} > D$ or $\sum_{t=1}^{r-1} c_{i_t, i_{t+1}} > L$.

These can be eliminated by partitioning the current infeasible subproblem, as in Step 3 of the TSP algorithm, taking $v(S) = 1$. More sophisticated partitioning schemes for $v(S) \geq 1$ are detailed in Laporte, Mercure and Nobert (1986).

In order to interpret the output as a CVRP solution, we apply the following rules. Consider an arc (i, j) belonging to the solution:

(i) if $i \in V \setminus \{1\}$ and $j \in V' \setminus V$, replace (i, j) by $(i, 1)$;

(ii) if $i \in V' \setminus V$ and $j \in V \setminus \{1\}$, replace (i, j) by $(1, j)$;

(iii) if $i, j \in V' \setminus V$, delete (i, j).

Using this methodology, Laporte, Mercure and Nobert (1986) have solved to optimality randomly generated asymmetrical CVRPs involving up to 260 vertices. Extensions to problems involving several types of side constraints are reported in Laporte, Mercure and Nobert (1991).

2.2. *The k-degree center tree and a related algorithm*

Christofides, Mingozzi and Toth (1981a) have developed an algorithm for symmetrical VRPs defined on a graph $G = (V, E)$. It is based on the following 'k-degree center tree relaxation' of the m-TSP where m is fixed. In any feasible solution, the set E of edges can be partitioned into four subsets:

E_0: edges not belonging to the solution;

E_1: edges forming a *k-degree center tree*, i.e. a spanning tree over G where the degree of vertex 1 is equal to k (with $k = 2m - y$);

E_2: y edges incident to vertex 1 ($0 \le y \le m$);

E_3: $m - y$ edges not incident to vertex 1.

In this representation, it is implicitly assumed that vehicle routes including vertex 1 and only one other vertex j are represented by 2 edges between 1 and j. The partition of E into four subsets is illustrated in Figure 1 for a 14-vertex problem, with $m = 4$. In this figure, $y = 3$ and $k = 2m - y = 5$.

Now, denote by l any edge of E and by c_l, its cost (length). Further define

E^i: the set of all edges incident to vertex i;

$(S, \bar{S})$: the set of all edges with one vertex in S and one vertex in $\bar{S}$,

$\xi_l^t (t = 1, 2, 3; l \in E)$: binary variables equal to 1 if and only if in the optimal solution, edge l belongs to E_t.

The problem is then

$$\text{(VRP2)} \quad \text{minimize} \quad \sum_{l \in E} c_l(\xi_l^1 + \xi_l^2 + \xi_l^3) \tag{7}$$

$$\text{subject to} \quad \sum_{l \in (S, \bar{S})} \xi_l^1 \ge 1 \quad (S \subset V;\ |S| \ge 1), \tag{8}$$

$$\sum_{l \in E^1} \xi_l^1 = 2m - y, \tag{9}$$

$$\sum_{l \in E} \xi_l^1 = n - 1, \tag{10}$$

$$\sum_{l \in E^1} \xi_l^2 = y, \tag{11}$$

$$\sum_{l \in E \setminus E^1} \xi_l^3 = m - y, \tag{12}$$

$$\sum_{l \in E_i} (\xi_l^1 + \xi_l^2 + \xi_l^3) = 2 \quad (i = 2, \dots, n), \tag{13}$$

$$\xi_l^1 \in \{0, 1\} \quad (l \in E), \tag{14a}$$

$$\xi_l^2 \in \{0, 1\} \quad (l \in E), \tag{14b}$$

$$\xi_l^3 \in \{0, 1\} \quad (l \in E), \tag{14c}$$

$$0 \le y \le m \quad \text{and integer.} \tag{15}$$

In this formulation, constraints (8)–(10) define a k-degree center tree; constraints (11) stipulate that there must be y additional edges (with respect to k) incident to the depot, and constraints (12) state there must be an additional $m - y$ edges not incident to the depot. Constraints (13) specify that the degree of every vertex, except the depot's, is equal to 2. The objective is the sum of all edge costs in the solution.

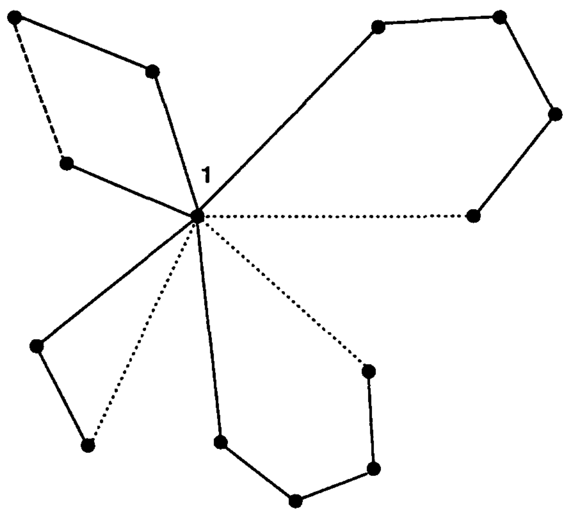

Figure 1. Partition of all edges of a feasible solution into subsets. ——— k-degree center tree ($k = 5$), ······ y edges adjacent to the depot ($y = 3$), - - - - - - $m - y = 1$ edges not adjacent to the depot

Constraints (13) can be relaxed in a Lagrangean fashion. Associate with each of these a multiplier λ_i and rewrite the objective as

$$\sum_{l\in E} c_l(\xi_l^1 + \xi_l^2 + \xi_l^3) + \sum_{l\in E} (\lambda_{\alpha(l)} + \lambda_{\beta(l)})(\xi_l^1 + \xi_l^2 + \xi_l^3) - 2\sum_{i=2}^{n} \lambda_i, \tag{16}$$

where $\lambda_1 = 0$ and $\alpha(l)$ $\beta(l)$ are the two terminal vertices of edge l. For a fixed y, (VRP2) can be rewritten as:

$$\begin{aligned} &\text{minimize} \quad \sum_{t=1}^{3} \sum_{l\in E} (c_l + \lambda_{\alpha(l)} + \lambda_{\beta(l)})\xi_l^t - 2\sum_{i=2}^{n} \lambda_i \\ &\text{subject to} \quad (8)\text{–}(14). \end{aligned} \tag{17}$$

The optimal value of (17) is not less than $\sum_{t=1}^{3} z^t(\lambda, y) - 2\sum_{i=2}^{n}\lambda_i$, where

$$\begin{aligned} z^1(\lambda, y) &= \min \sum_{l\in E} (c_l + \lambda_{\alpha(l)} + \lambda_{\beta(l)})\xi_l^1 \\ &\quad \text{subject to (8)–(10) and (14a);} \\ z^2(\lambda, y) &= \min \sum_{l\in E} (c_l + \lambda_{\alpha(l)} + \lambda_{\beta(l)})\xi_l^2 \\ &\quad \text{subject to (11) and (14b), and} \\ z^3(\lambda; y) &= \min \sum_{l\in E} (c_l + \lambda_{\alpha(l)} + \lambda_{\beta(l)})\xi_l^3 \\ &\quad \text{subject to (12) and (14c).} \end{aligned}$$

For given λ and y, the problems of determining $z^t(\lambda, y)$ for $t = 1, 2, 3$ are easy and can be solved in polynomial time.

A lower bound on the optimal VRP solution is then given by

$$\max_{m_1 \le y \le m} \max_{\lambda} \left\{ \sum_{t=1}^{3} z^t(\lambda, y) - 2\sum_{i=2}^{n} \lambda_i \right\}, \tag{18}$$

where m_1 is a lower bound on the number of routes made up of the depot and only one vertex in $V \backslash \{1\}$. Taking into account capacity- and maximum-length constraints, m_1 can be determined so as to satisfy the following conditions:

(i) suppose the vertices are ordered in decreasing order of their weight d_i. Then m_1 is the largest value satisfying

$$(m - m_1)D \geq \sum_{i=m_1+1}^{n} d_i;$$

(ii) similarly, every vertex contributes an amount of at least $u_i = \delta_i + (c_{ii_1} + c_{ii_2})$ to the length of a route, where δ_i is the service time of vertex i and i_1,i_2 are the two vertices nearest to i. Then, if the vertices are ordered in decreasing order of the u_i''s, m_1 must satisfy

$$(m - m_1)L \geq \sum_{i=m_1+1}^{n} u_i.$$

Christofides, Mingozzi and Toth (1981a) have embedded the lower bound defined by (18) in a branch-and-bound scheme and have successfully solved VRPs ranging in size from 10 to 25 vertices.

2.3. *Dynamic programming*

Dynamic programming was first proposed for VRPs by Eilon, Watson-Gandy and Christofides (1971). Consider a VRP with a fixed number m of vehicles. Let $c(S)$ denote the cost (length) of a vehicle route through vertex 1 and all vertices of a subset S of $V\setminus\{1\}$. Let $f_k(U)$ be the minimum cost achievable using k vehicles and delivering to a subset U of $V\setminus\{1\}$. Then the minimum cost can be determined through the following recursion:

$$f_k(U) = \begin{cases} c(U) & (k=1), \\ \min\limits_{U^* \subset U \subseteq V\setminus\{1\}} [f_{k-1}(U\setminus U^*) + c(U^*)] & (k>1). \end{cases} \tag{19}$$

The solution cost is equal to $f_m(V\setminus\{1\})$ and the optimal solution corresponds to the optimizing subsets U^* in (19).

It is apparent that if $f_k(U)$ has to be computed for all values of k and for all subsets U of $V\setminus\{1\}$, the number of computations required is likely to be excessive in most problems. Efficient use of dynamic programming requires a substantial reduction of the number of states by means of a relaxation procedure, or by using feasibility or dominance criteria. For example, in the CVRP the sets U and U^* must satisfy

$$\sum_{i\in V\setminus\{1\}} d_i - (m-k)D \leq \sum_{i\in U} d_i \leq kD \quad (k=1,\dots,m) \tag{20}$$

and

$$\sum_{i\in U} d_i - (k-1)D \leq \sum_{i\in U^*} d_i \leq D \quad (k=1,\dots,m). \tag{21}$$

State-space relaxation provides another efficient way of reducing the number of states. The method was introduced by Christofides, Mingozzi and Toth (1981b). It provides a longer bound on the cost of the optimal solution. The optimum can then be reached by embedding the bounding procedure in an enumerative scheme. The method can be summarized as follows. Consider the general DP recursion

$$f_{0,i}(0, j) = \min_{k\in\Delta^{-1}(j)} [f_{0,i-1}(0, k) + c_i(k, j)] \tag{22}$$

where
$f_{0,i}(0, j)$ is the least cost of going from state 0 at stage 0 to state j at stage i,
$\Delta^{-1}(j)$ is the set of all possible states from which state j can be reached directly, and $c_i(k, j)$ is the cost of going from state k at stage $i-1$ to state j at stage i.
Let $g(\cdot)$ be a mapping from the state space S associated with (22) to a state space T of smaller cardinality, and let $F^{-1}(g(j))$ be a set satisfying

$$k \in \Delta^{-1}(j) \Rightarrow g(k) \in F^{-1}(g(j)). \tag{23}$$

Recursion (22) then becomes

$$f_{0,i}(g(0), g(j)) = \min_{t \in F^{-1}(g(j))} [f_{0,i-1}(g(0), t) + \bar{c}_i(t, g(j))] \tag{24}$$

where

$$\bar{c}_i(t, g(j)) = \min[c_i(k, l) : g(k) = t, g(j) = g(l)]. \tag{25}$$

It results that

$$f_{0,i}(g(0), g(i)) \le f_{0,i}(0, i).$$

This relaxation is useful only if

(i) $F^{-1}(\cdot)$ can easily be determined: this will be so if $g(\cdot)$ is *separable*, so that given $g(U)$ and r, $g(U \setminus \{r\})$ can be computed;

(ii) $g(\cdot)$ is such that the optimization of (25) is over a small domain or that a good lower bound on $\bar{c}_i(t, g(j))$ can be computed.

Christofides, Mingozzi and Toth (1981b) have used the following relaxation for CVRPs. Let $f_k(U, r)$ be the least cost of supplying a set U of vertices, using k vehicles, where the last vertices of the k corresponding routes belong to $\{2, \dots, r\}$ $(k \le r \le n)$. Let $c(U, r)$ be the cost of the TSP solution through $U \cup [1\}$, where the last vertex before the depot is r. The recursion is then

$$f_k(U, r) = \begin{cases} \min\left[f_k(U, r-1), \min_{u^* \in U} \{f_{k-1}(U \setminus U^*, r-1) + c(U^*, r)\}\right] & (k, r > 1), \\ c(U, r) & (k = 1), \end{cases} \tag{26}$$

subject to (20). For this problem, the mapping function is given by

$$g(U) = \sum_{i \in U} d_i.$$

Recursion (24) then becomes

$$f_k(g(U), r) = \min\left[f_k(g(U), r-1), \min_p \{f_{k-1}(g(U) - p, r-1) + \bar{c}(p, r)\}\right], \tag{27}$$

$$\text{subject to } g(V) - (m-1)D \le p \le \min(g(U), D). \tag{28}$$

Using this and other relaxations, lower bounds on optimal VRP solutions were obtained for 10 problems containing 10 to 25 vertices. The ratio 'lower bound/optimum' varied between 93.1% and 100%. More recently, Christofides (1985b) reported that CVRPs with up to 50 vertices could be solved systematically with this approach.

2.4. *Set partitioning and column generation*

Balinski and Quandt (1964) were among the first to propose a set partitioning formulation for VRPs. Consider J, the set of all feasible routes j and a_{ij} be a binary coefficient equal to 1 if and only if vertex $i > 1$ appears on route j. Let c_j^* be the optimal cost of route j and x_j, a binary variable equal to 1 if and only if route j is used in the optimal solution.

The problem can then be formulated as follows:

(VRP3) minimize $$\sum_{j \in J} c_j^* x_j \tag{29}$$

subject to $$\sum_{j \in J} a_{ij} x_j = 1 \quad (i \in V \setminus \{1\}), \tag{30}$$

$$x_j \in \{0, 1\} \ (j \in J). \tag{31}$$

There are two main difficulties associated with this formulation:

(i) the large number of binary variables x_j which can run into the millions in most real-life cases. Only in extremely constrained problems (i.e. in problems with very few feasible solutions) will the number of variables be small enough to enable the problem to be solved directly;

(ii) the difficulty of computing the c_j^* values. For example, in the CVRP, every route j corresponds to a set of vertices S_j satisfying

$$\sum_{i \in S_j} d_j \leq D.$$

The value of c_j^* is then obtained by solving a TSP on S_j.

However, if the number of variables is relatively small and the objective is to minimize the number of vehicles, i.e. $c_j^* = 1$ for all $j \in J$, the linear relaxation of (VRP3) often provides an integer solution (Toregas and ReVelle, 1972). If the solution (x^*) is non-integer and gives a fractional objective value, then the cutting plane

$$\sum_{j \in J} x_j > \left\lceil \sum_{j \in J} x_j^* \right\rceil$$

can be introduced. Very few cuts are generally required to reach integrality (Orloff, 1976).

A natural way around the difficulties just mentioned is to use a *column generation algorithm*. This technique has been applied to the field of vehicle routing by Rao and Zionts (1968), Foster and Ryan (1976), Orloff (1976), Desrosiers, Soumis and Desrochers (1984), Agarwal, Mathur and Salkin (1989), and Desrochers, Desrosiers and Solomon (1990). A recent introduction to the field can be found in Haouari, Dejax and Desrochers (1990).

In column generation, a *reduced problem* containing only a restricted subset of all possible columns (variables) is repeatedly solved. The linear relaxation of the reduced problem provides an optimal dual variable vector λ. Checking for optimality implies computing the column s of least marginal cost, i.e., determining the column s satisfying

$$c_s^* - \lambda y_s = \min_{j \in J} \{c_j^* - \lambda y_j\} \tag{32}$$

where y_j is the column-vector of constraint coefficients of variable x_j. If the marginal cost of x_s is non-negative, the current solution is optimal and the procedure terminates. Otherwise, x_s enters the basis and the problem is re-optimized. Since VRP solutions must be integer, this procedure must be used in conjunction with a branch-and-bound algorithm. Solving (32) is done using an algorithm for a shortest-path problem constrained in the same manner as the original VRP. This procedure has been applied with success by Desrosiers, Soumis and Desrochers (1984) and more recently, by Desrochers, Desrosiers and Solomon (1991) to the solution of VRPs with time windows containing up to 100 vertices. As expected, the method performs better on tightly constrained problems, as the number of feasible columns is then smaller.

2.5. *A three-index vehicle flow formulation*

Fisher and Jaikumar (1978, 1981) have developed a three-index vehicle flow formulation for VRPs with capacity restrictions, time windows and no stopping times δ_i. Such formulations use variables to represent the passing of a vehicle on an arc or edge (i, j). In three-index formulations, variables x_{ijk} indicate whether (i, j) is traversed by vehicle k or not. In two-index formulations, variables x_{ij} do not specify which vehicle is used on (i, j). Fisher and Jaikumar have also developed an algorithm based on this formulation. Although this algorithm seems to have been used only to provide a heuristic solution to the problem, it guarantees an optimal solution in a finite number of steps, if run to completion. The formulation does not require vehicles to be identical. Let D_k be the capacity of vehicle k, $[a_i, b_i]$ the time window for vertex i, and t_{ij}, the travel time on arc (i, j). Define binary variables x_{ijk} $(i \neq j)$, equal to 1 if and only if in the optimal solution, arc (i, j) is traversed by vehicle k. Also define binary variables y_{ik}, equal to 1 if and only if vertex i is served by vehicle k. Finally, let t_i denote the arrival time at vertex i and let T be a very large number.

The formulation is then

$$\text{(VRP4)}\quad \text{minimize} \quad \sum_{k=1}^{n} \sum_{i \neq j} c_{ij} x_{ijk} \tag{33}$$

$$\text{subject to} \quad \sum_{i=1}^{n} d_i y_{ik} \leq D_k \quad (k = 1, \dots, m), \tag{34}$$

$$\sum_{k=1}^{m} y_{ik} = \begin{cases} m & (i = 1), \\ 1 & (i = 2, \dots, n), \end{cases} \tag{35}$$

$$\sum_{i=1}^{n} x_{ijk} = y_{jk} \quad (j = 1, \dots, n;\ k = 1, \dots, m), \tag{36}$$

$$\sum_{j=1}^{n} x_{ijk} = y_{jk} \quad (j = 1, \dots, n;\ k = 1, \dots, m), \tag{37}$$

$$\sum_{i,j \in S} x_{ijk} \leq |S| - 1 \quad (S \subset V;\ |S| \geq 2;\ k = 1, \dots, m), \tag{38}$$

$$t_j \begin{cases} \geq t_i + t_{ij} - (1 - x_{ijk})T \\ \leq t_i + t_{ij} + (1 - x_{ijk})T \end{cases} \quad (i, j = 1, \dots, n;\ k = 1, \dots, m), \tag{39}$$

$$a_i \leq t_i \leq b_i \quad (i = 2, \dots, n), \tag{40}$$

$$x_{ijk} \in \{0, 1\} \quad (i, j = 1, \dots, n;\ k = 1, \dots, m), \tag{41}$$

$$y_{ik} \in \{0, 1\} \quad (i = 1, \dots, n;\ k = 1, \dots, m). \tag{42}$$

Most constraints of this formulation are either self-explanatory or have been previously discussed. Note that if arc (i, j) does not appear in the solution, x_{ijk} is equal to 0 for all k and constraints (39) are ineffective then; otherwise, x_{ijk} is equal to 1 for some k and then $t_j = t_i + t_{ij}$.

Essentially, two well known problems are contained in (VRP4):

(i) the *generalized assignment problem* (GAP) obtained by relaxing constraints (38), (39) and (41);

(ii) the *TSP with time windows* (TSPTW): for given y_{ik}'s satisfying the GAP constraints and for a given k, constraints (36)–(40) are those of a TSPTW for vehicle k.

Fisher and Jaikumar propose an algorithm based on Benders' decomposition (Benders, 1962). The procedure iterates between solving a GAP master problem that assigns vertices to vehicles, and solving a TSPTW to determine the best vehicle route for each vehicle. The method has the advantage of producing a feasible solution, even if not run to completion (its authors did not, in fact, run it to optimality). Also, since it repeatedly solves a GAP and a TSPTW, it can benefit directly from any improvement in algorithms for these two problems. Such improvements have been proposed recently by Martello and Toth (1990) and by Desrochers, Desrosiers and Solomon (1991). Fisher and Jaikumar (1981) report computational results for VRPs ranging from 50 to 199 vertices.

2.6. *A two-index vehicle flow formulation*

In symmetrical CVRPs and DVRPs, a more compact formulation can be obtained by dropping the index k from the variables. Here x_{ij} $(i < j)$ indicates how many timed edge (i, j) is traversed by a vehicle: if $i, j \in V \backslash \{1\}$, then $x_{ij} \in \{0, 1\}$; if $i = 1$ and $j \in V \backslash \{1\}$, then $x_{ij} \in \{0, 1, 2\}$. The case $x_{ij} = 2$ corresponds to the single vertex trip $(1, j, 1)$. In addition,

(i) in CVRPs, x_{ij} is not defined if $d_i + d_j > D$;

(ii) in DVRPs, let s_i be the length of a shortest path from vertex 1 to vertex i and t_j, the length of a shortest path from vertex j to vertex 1. Then x_{ij} is not defined whenever $s_i + c_{ij} + t_j > L$.

The following formulation was proposed by Laporte, Nobert and Desrochers (1985).

$$\text{(VRP5)} \quad \text{minimize} \quad \sum_{i<j}^{n} c_{ij} x_{ij} \tag{43}$$

$$\text{subject to} \quad \sum_{j=2}^{n} x_{1j} = 2m, \tag{44}$$

$$\sum_{i<k} x_{ik} + \sum_{j>k} x_{kj} = 2 \quad (k = 2,\ldots,n), \tag{45}$$

$$\sum_{\substack{i,j\in S\\ i<j}} x_{ij} \le |S| - v(S) \quad (S \subset V\setminus\{1\};\ 2 \le |S| \le n-2), \tag{46}$$

$$x_{1j} \in \{0, 1, 2\} \quad (j = 2,\ldots,n), \tag{47}$$

$$x_{ij} \in \{0, 1\} \quad (i, j = 2,\ldots,n). \tag{48}$$

This formulation is a direct extension of the corresponding (SYM) formulation for the TSP (Laporte, 1991). It is worth noting that m can be taken as a fixed constant or as a variable. In the latter case, it is often convenient to impose a lower or an upper bound on m. In subtour elimination constraints (46), $v(S)$ is a lower bound on the number of vehicles required to visit S. As in the TSP, the validity of these constraints can easily be established from the identity

$$\sum_{k\in S}\left(\sum_{i<k} x_{ik} + \sum_{j>k} x_{kj}\right) = 2\sum_{\substack{i,j\in S\\ i<j}} x_{ij} + \sum_{\substack{i\in S, j\in \bar{S}\\ \text{or } i\in\bar{S}, j\in S}} x_{ij}$$

and the connectivity constraints

$$\sum_{\substack{i\in S, j\in \bar{S}\\ \text{or } i\in\bar{S}, j\in S}} x_{ij} \ge 2v(S) \quad (S \subset V\setminus\{1\};\ 3 \le |S| \le n-3). \tag{46a}$$

These constraints force at least $2v(S)$ edges to exist between S and its complement if $v(S)$ vehicles are used for S. As in the asymmetrical case (VRP1), the value of $v(S)$ can be taken as $\lceil(\Sigma_{i\in S} d_i)/D\rceil$ for CVRPs, but is not so easily computed for DVRPs (see Laporte, Nobert and Desrochers, 1985).

This model can be solved by means of the following constraint relaxation algorithm:

Step 1. (Initialization) Define a first subproblem containing (43), the degree constraints (44) and (45), as well as the bounds on the variables. Insert this subproblem in a queue. Let z^* be the value of the best known VRP solution (If no such solution is known, set $z^* := \infty$.)

Step 2. (Termination check) If the queue is empty, stop. Otherwise, extract from it the next subproblem to be solved.

Step 3. (Subproblem solution) Solve the subproblem (a linear program). Let $\bar{z}$ be its solution value.

Step 4. (Feasibility check) If $\bar{z} \ge z^*$, fathom the current subproblem and go to Step 2. Otherwise, check whether any violated subtour elimination constraint can be identified. If so, generate one or several such constraints and go to Step 3. If no violation of constraints (46) can be detected, check whether the solution is integer. If it is, set $z^* := \bar{z}$ and go to Step 2. If the solution is non-integer, create subproblems by branching on a fractional variables; insert these subproblems in the queue and go to Step 2.

This algorithm has been used by Laporte, Nobert and Desrochers (1985) on problems containing up to 60 vertices. Loosely constrained problems are easier than tight problems since they require fewer subtour elimination constraints. By contrast, the k-degree centre-tree algorithm, dynamic programming and set partitioning work better on tightly constrained problems.

3. Heuristic algorithms

Heuristic algorithms for the VRP can often be derived from procedures derived from the TSP (see Laporte, 1992, Section 5.2). The nearest neighbour algorithm, insertion algorithms and tour improvement procedures can be applied to CVRPs and DVRPs almost without modifications. However, when applying these methods to VRPs care must be taken to ensure that only feasible vehicle routes are created. The Fisher and Jaikumar (1978) algorithm described in Section 2.5 may also be viewed as a heuristic since it provides at every step a feasible VRP solution and is often interrupted before optimality can be achieved.

In this section, we will describe four heuristics specifically developed for the VRP.

3.1. The Clarke and Wright algorithm (1964)

This classical algorithm was first proposed in 1964 by Clarke and Wright to solve CVRPs in which the number of vehicles is free. The method starts with vehicle routes containing the depot and one other vertex. At each step, two routes are merged according to the largest saving that can be generated.

Step 1. Compute the savings $s_{ij} = c_{i1} + c_{1j} - c_{ij}$ for $i, j = 2, \ldots, n$, and $i \neq j$. Create $n - 1$ vehicle routes $(1, i, 1)$ $(i = 2, \ldots, n)$.

Step 2. Order the savings in a non-increasing fashion.

Step 3. Consider two vehicle routes containing arcs $(i, 1)$ and $(1, j)$, respectively. If $s_{ij} > 0$, tentatively merge these routes by introducing arc (i, j) and by deleting arcs $(i, 1)$ and $(1, j)$. Implement the merge if the resulting route is feasible. Repeat this step until no further improvement is possible. Stop.

This procedure can be executed in $O(n^2 \log n)$ time, but this complexity can be reduced by using appropriate data structures (Golden et al., 1977; Nelson et al., 1985; Paessens, 1988). Gaskell (1967), Yellow (1970) and Paessens (1988) have also proposed a number of variants of this method. The Clarke and Wright algorithm implicitly ignores vehicle fixed costs and fleet size. Vehicle costs f can easily be taken into account by adding this constant to every c_{1j} $(j = 2, \ldots, n)$. Solutions with a fixed number of vehicles can be obtained by repeating Step 3 until the required number of routes has been reached, even if the savings become negative.

3.2. The sweep algorithm (Wren, 1971; Wren and Holliday, 1972; Gillett and Miller, 1974)

It seems that the origins of the sweep algorithm can be traced back to the work of Wren (1971) and Wren and Holliday (1972) for CVRPs with one or several depots, and vertices located in the Euclidean plane. The method is commonly attributed to Gillett and Miller (1974) who gave it its name. In order to ease the implementation of this method, it is preferable to represent vertices by their polar coordinates (θ_i, ρ_i), where θ_i is the angle and ρ_i is the ray length. Assign a value $\theta_i^* = 0$ to an arbitrary vertex i^* and compute the remaining angles from $(1, i^*)$. Rank the vertices in increasing order of their θ_i. A possible implementation of the method is the following.

Step 1. Choose an unused vehicle k.

Step 2. Starting from the unrouted vertex having the smallest angle, assign vertices to the vehicle as long as its capacity is not exceeded. If unrouted vertices remain, go to Step 1.

Step 3. Optimize each vehicle route separately by solving the corresponding TSP (exactly or approximately). Perform vertex exchanges between adjacent routes if this saves distance. Re-optimize and stop.

3.3. The Christofides-Mingozzi-Toth two-phase algorithm (1979)

This algorithm was basically designed for CVRPs and DVRPs. It produces two alternative solutions for given parameters $\lambda \geq 1$ and $\mu \geq 1$ set by the user. The better of the two solutions can then be selected. This procedure can be repeated for several values of λ and μ.

Phase 1 Sequential route construction.

Step 1. Set a first route index k equal to 1.

Step 2. Select any unrouted vertex i_k to initialize route k. For every unrouted vertex i, compute $\delta_i = c_{1i} + \lambda c_{ii_k}$.

Step 3. Let $\delta_{i*} = \min_{i \in S_k}\{\delta_i\}$, where S_k is the set of unrouted vertices that can be feasibly inserted into route k. Insert vertex $i*$ into route k. Optimize route k using an r-opt algorithm (Lin, 1965). Repeat Step 3 until no more vertices can be assigned to route k.

Step 4. If all vertices have been inserted into routes, stop. Otherwise, set $k := k + 1$ and go to Step 2.

Phase 2 Parallel route construction.

Step 5. Initialize k routes $R_t = (1, i_t, 1)$ $(t = 1, \ldots, k)$, where k is the number of routes obtained at the end of Phase 1. Let $K = \{R_1, \ldots, R_k\}$.

Step 6. For each route $R_t \in K$ and for each vertex i not yet associated with a route, compute $\varepsilon_{ti} = c_{1i} + \mu c_{1k_t} - c_{1i_t}$ and $\varepsilon_{t*i} = \min_t\{\varepsilon_{ti}\}$. Associate vertex i with route R_{t*} and repeat Step 6 until all vertices have been associated with a route.

Step 7. Take any route $R_t \in K$ and set $K := K \backslash \{R_t\}$. For every vertex i associated with route R_t, compute $\varepsilon_{t'i} = \min_{R_t \in K}\{\varepsilon_{t_i}\}$ and $\tau_i = \varepsilon_{t'i} - \varepsilon_{ti}$.

Step 8. Insert into route R_t vertex $i*$ satisfying $\tau_{i*} = \max_{i \in S_t}\{\tau_i\}$, where S_t is the set of unrouted vertices associated with route R_t that can feasibly be inserted into route R_t. Optimize route R_t using an r-opt algorithm. Repeat Step 8 until no more vertices can be inserted into route R_t.

Step 9. If $K \neq \emptyset$, go to Step 6. Otherwise, if all vertices are routed, stop. If unrouted vertices remain, create new routes starting with Step 1 of Phase 1.

3.4. A tabu search algorithm (Gendreau, Hertz and Laporte, 1991)

This tabu search heuristic (see Laporte, 1992, Section 5.2.2) constructs a sequence of solutions, and then executes an improvement step. The successive vehicle routes produced by the algorithm may not be feasible; their degree of departure from feasibility is measured by means of a penalty in the objective function.

Step 1. Constitute back and forth routes between the depot and customers. Let x be this solution and let its cost be $F(x)$. Set the tabu list $T := \varnothing$.

Step 2. Define $N(x)$, the neighbourhood of x, as the set of all solutions that can be reached by inserting an arbitrary vertex in its p-neighbourhood by using the GENI procedure (Gendreau, Hertz and Laporte, 1992). If $N(x) \backslash T = \emptyset$, go to Step 3. Otherwise, identify the least cost solution y in $N(x) \backslash T$ and set $x := y$. Update the best known solution.

Step 3. If the maximum number of iterations since the beginning of the process or since the last update has been reached, go to Step 4. Otherwise, update T and go to Step 2.

Step 4. Attempt to improve each of the vehicle routes by means of the US post-optimization procedure (Gendreau, Hertz and Laporte, 1992).

This algorithm was successfully applied to a number of classical VRPs described in the OR literature. Computational results indicate that the proposed heuristic may be one of the best ever developed for the VRP

4. Conclusion

The Vehicle Routing Problem lies at the heart of distribution management. There exist several versions of the problem, and a wide variety of exáct and approximate algorithms have been proposed for its solution. Exact algorithms can only solve relatively small problems, but a number of approximate algorithms have proved very satisfactory. However, several promising avenues of research deserve more attention, such as tabu search methods.

Acknowledgements

This research was in part supported by the Canadian Natural Sciences and Engineering Research Council (grant OGP0039682), and by a Quebec-New-Brunswick interprovincial research grant. An extended version of this paper will appear in *Discrete Optimization Models*, H.A. Eiselt and C.L. Sandblom (eds.), De Gruyter, Berlin, 1993. Permission to publish this paper in EJOR is gratefully acknowledged. Finally, thanks are also due to Paolo Toth for his valuable comments on a preliminary version of this paper.

References

Agarwal, Y., Mathur, K., and Salkin, H.M. (1989), "A set-partitioning-based algorithm for the vehicle routing problem", *Networks* 19, 731–750.

Balinski, M., and Quandt, R. (1964), "On an integer program for a delivery problem", *Operations Research* 12, 300–304.

Benders, J.F. (1962), "Partitioning procedures for solving mixed-variables programming problems", *Numerische Mathematik* 4, 238–252.

Bodin, L.D., Golden, B.L., Assad, A.A., and Ball, M.O. (1983), "Routing and scheduling of vehicles and crews. The state of the art", *Computers and Operations Research* 10, 69–211.

Christofides, N. (1985a), "Vehicle routing", in: E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan and D.B. Shmoys, (eds.), *The Traveling Salesman Problem. A Guided Tour of Combinatorial Optimization*, Wiley, Chichester, 431–448.

Christofides, N. (1985b), "Vehicle scheduling and routing", Presented at the *12th International Symposium on Mathematical Programming*, Cambridge, MA.

Christofides, N., Mingozzi, A., and Toth, P. (1979), "The vehicle routing problem", in: N. Christofides, A. Mingozzi, P. Toth and C. Sandi (eds.), *Combinatorial Optimization*, Wiley, Chichester, 315–338.

Christofides, N., Mingozzi, A., and Toth, P. (1981a), "Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations", *Mathematical Programming* 20, 255–282.

Christofides, N., Mingozzi, A., and Toth, P. (1981b), "State space relaxation procedures for the computation of bounds to routing problems", *Networks* 11, 145–164.

Clarke, G., and Wright, J.W. (1964), "Scheduling of vehicles from a central depot to a number of delivery points", *Operations Research* 12, 568–581.

Desrochers, M., Desrosiers, J., and Solomon, M.M. (1991), "A new optimization algorithm for the vehicle routing problem with time windows", *Operations Research*, forthcoming.

Desrochers, M., Lenstra, J.K., and Savelsbergh, M.W.P. (1990), "A classification scheme for vehicle routing and scheduling problems", *European Journal of Operational Research* 46, 322–332.

Desrosiers, J., Soumis, F., and Desrochers, M. (1984), "Routing with time windows by column generation", *Networks* 14, 545–565.

Eilon, S., Watson-Gandy, C.D.T., and Christofides, N. (1971), *Distribution Management: Mathematical Modelling and Practical Analysis*. Griffin, London.

Fisher, M.L., and Jaikumar, R. (1978), "A decomposition algorithm for large-scale vehicle routing", Working Paper 78-11-05, Department of Decision Sciences, University of Pennsylvania.

Fisher, M.L., and Jaikumar, R. (1981), "A generalized assignment heuristic for vehicle routing", *Networks* 11, 109–124.

Foster, B., and Ryan, D. (1976), "An integer programming approach to the vehicle scheduling problem", *Operational Research Quarterly* 27, 367–384.

Gaskell, T. (1967), "Bases for vehicle fleet scheduling", *Operational Research Quarterly* 18, 281–295.

Gendreau, M., Hertz, A., and Laporte, G. (1991), "A tabu search heuristic for the vehicle routing problem", Publication #777, Centre de recherche sur les transports, Montréal.

Gendreau, M., Hertz, A., and Laporte, G. (1992), "New insertion and post-optimization procedures for the traveling salesman problem", *Operations Research* forthcoming.

Gillett, B., and Miller, L. (1974), "A heuristic algorithm for the vehicle dispatch problem", *Operations Research* 22, 340–349.

Golden, B.L., and Assad, A.A. (1988), *Vehicle Routing: Methods and Studies*, North-Holland, Amsterdam.

Golden, B.L., Magnanti, T.L., and Nguyen, H.Q. (1977), "Implementing vehicle routing algorithms", *Networks* 7, 113–148.

Haouari, M., Dejax, P.J., and Desrochers, M. (1990), "Modelling and solving complex vehicle routing problems using column generation", Working Paper, LEIS, École Centrale de Paris.

Laporte, G. (1990), "Développements algorithmiques récents et perspectives de recherche en distributique", *Les Cahiers Scientifiques du Transport* 21, 61–84.

Laporte, G. (1992), "The traveling salesman problem: An overview of exact and approximate algorithms", *European Jouranl of Operational Research* 59/2, 231–248.

Laporte, G., Mercure, H., and Nobert, Y. (1986), "An exact algorithm for the asymmetrical capacitated vehicle routing problem", *Networks* 16, 33–46.

Laporte, G., Mercure, H., and Nobert, Y. (1991), "A branch-and-bound algorithm for a class of asymmetrical vehicle routing problems", *Journal of the Operational Research Society*, forthcoming.

Laporte, G., and Nobert, Y. (1987), "Exact algorithms for the vehicle routing problem", in: S. Martello, G. Laporte, M. Minoux and C. Ribeiro (eds.), *Surveys in Combinatorial Optimization*, North-Holland, Amsterdam, 147–184.

Laporte, G., Nobert, Y., and Desrochers, M. (1985), "Optimal routing under capacity and distance restrictions", *Operations Research* 33, 1050–1073.

Lenstra, J.K., and Rinnooy Kan, A.H.G. (1975), "Some simple applications of the travelling salesman problem", *Operational Research Quarterly* 26, 717–734.

Lin, S. (1965), "Computer solutions of the traveling salesman problem", *Bell System Technical Journal* 44, 2245–2269.

Martello, S., and Toth, P. (1990), "Generalized assignment problem", in: S. Martello and P. Toth (eds.), *Knapsack Problems. Algorithms and Computer Implementations*, Wiley, Chichester, 189–220.

Nelson, M.D., Nygard, K.E., Griffin, J.H., and Shreve, W.E. (1985), "Implementation techniques for the vehicle routing problem", *Computers & Operations Research* 12, 273–283.

Orloff, C. (1976), "Route-constrained fleet scheduling", *Transportation Science* 10, 149–168.

Paessens, H. (1988), "The savings algorithm for the vehicle routing problem", *European Journal of Operational Research* 34, 336–344.

Rao, M.R., and Zionts, S. (1968), "Allocation of transportation units to alternative trips– A column generation scheme with out-of-kilter subproblems", *Operations Research* 16, 52–63.

Toregas, C., and ReVelle, C. (1972), "Location under time or distance constraints", *Papers of the Regional Science Association* 28, 133–143.

Wren, A. (1971), *Computers in Transport Planning and Operation*, Ian Allan, London.

Wren, A., and Holliday, A. (1972), "Computer scheduling of vehicles from one or more depots to a number of delivery points", *Operations Research Quarterly* 23, 333–344.

Yellow, P. (1970), "A computational modification to the savings method of vehicle scheduling", *Operational Research Quarterly* 21, 281–283.