REVUE FRANÇAISE D'AUTOMATIQUE, INFORMATIQUE, RECHERCHE OPÉRATIONNELLE. RECHERCHE OPÉRATIONNELLE

NICOS CHRISTOFIDES

The vehicle routing problem

Revue française d'automatique, informatique, recherche opérationnelle. Recherche opérationnelle, tome 10, n° V1 (1976), p. 55-70

<http://www.numdam.org/item?id=RO_1976__10_1_55_0>

R.A.I.R.O. Recherche opérationnelle
(vol. 10, n° 2, février 1976, p. 55 à 70)

THE VEHICLE ROUTING PROBLEM (*)

by Nicos CHRISTOFIDES (1)

Abstract. — *A set of customers with known locations and known requirements for some commodity, is to be supplied from a single depot by delivery vehicles of known capacity. The problem of designing routes for these vehicles so as to minimise the cost of distribution is known as the vehicle routing problem (VRP). In this paper we categorise, discuss and extend both exact and approximate methods for solving VRP's, and we give some results on the properties of feasible solutions which help to reduce the computational effort involved in solving such problems.*

1. INTRODUCTION

The *vehicle routing problem* (VRP) is a generic name given to a whole class of problems involving the visiting of "customers" by "vehicles". These problems derive their name from the basic practical problem of supplying geographically dispersed customers with goods using a number of vehicles operating from a common goods depot. The problem is one of routing the vehicles so that the total distance travelled (or time taken, cost incurred, etc.), is minimal. The VRP (also known in the literature as the "vehicle scheduling" [11, 13, 14] "vehicle dispatching" [6, 12, 15, 25] or simply as the "delivery" problem [2, 16, 26]) appears very frequently in practical situations not directly related to the delivery of goods. For example, the collection of mail from mail-boxes, the pickup of children by school buses, house-call tours by a doctor, preventive maintenance inspection tours, the delivery of laundry, etc. are all VRP's in which the "delivery" operation may be a collection, collection and/or delivery, or neither; and in which the "goods" and "vehicles" can take a variety of forms some of which may not even be of a physical nature. In view of the enormous number of practical situations which give rise to VRP's it is not surprising to find that an equally large number of constraints and/or objectives appear in such problems. However, the basic VRP can be characterised as follows. (Other characterisations can be found in [5].)

(*) Reçu mars 1975.

(1) Imperial College of Science and Technology, Department of Management Science, Londres.

1.1. The problem

Let $X = \{ x_i \mid i = 1, \ldots, N \}$ be a set of N customers and l_i be the location of customer x_i. A set of vehicles $Y = \{ y_k \mid k = 1, \ldots, M \}$ is given, stationed at a depot x_0 at location l_0. A customer x_i has the following basic requirements:

C1. A quantity q_i to be delivered.

C2. A time w_i required to unload the quantity q_i.

C3. A set of time periods during which delivery must be made.

The p-th time period is defined by the two times $B(p, i)$ and $A(p, i)$ before and after which (respectively) the delivery to customer x_i can begin.

C4. A subset $Y_i \subseteq Y$ of vehicles that can deliver to customer x_i.

A vehicle y_k has the following basic characteristics or requirements:

V1. A capacity Q_k for carrying goods.

V2. A total working time of T_k hours from departure to arrival back at the depot.

V3. A time B_k before, and a time A_k after which the vehicle must depart from the depot.

There are two basic objectives in the VRP:

O1. Find optimal routes to be operated by available vehicles so as to supply the customer requirements at minimum total variable cost.

or

O2. Find the smallest possible number of vehicles and their routes which can supply all customer requirements.

Since the cost of vehicles in the vehicle fleet is usually considered as a fixed cost, objectives O1 and O2 correspond to the minimisation of variable and of fixed costs respectively (assuming that all vehicles have the same fixed cost). Quite obviously, the minimum variable cost routes do not in general utilise the smallest possible number of vehicles although on occation the two answers may coinside [13].

In the above description of the VRP it is assumed that given two locations l_i and l_j, the distance $d(i, j)$, time of travel $t(i, j)$ or total variable cost $c(i, j)$ of going from l_i to l_j can be calculated for any i and $j = 0, \ldots, N$. If the locations are given in terms of coordinates, the distances can be calculated as the Euclidean of rectangular distances (whichever is appropriate). If the locations are on a road network they are calculated as the shortest distances. Similarly, the times $t(i, j)$ may be calculated taking account of different speeds in different regions or parts of the road network. In any case we will

use $c(i, j)$ to be the total variable "cost" and which may represent time, distance, or some other composite measure.

The VRP defined above is by necessity a gross simplification of problems found in practice. The most important ommissions which almost certainly must be included in any computer code for solving practical VRP' are [10, 19]:

(i) Multiple trips operated by vehicles in a single work period.

(ii) Multiple periods (e. g. many days with intermediate overnight stops) needed for certain vehicle trips.

(iii) The VRP is most often encountered not as a single period (day) problem, but as a problem over many periods (e. g. a week) with some customers requiring delivery once a week others twice of three times a week, etc. In such multiperiod VRP's the days on which a certain customer requires delivery may be completely unspecified or a combination of days is to be chosen from a number of acceptable ones. Thus, for any one day, the set X of customers to be delivered to must be decided from the total set of customers that must be supplied during the whole period.

In this paper we categorise, discuss and extend both exact and approximate methods for solving VRP's, and we give some results on the properties of feasible solutions which help to reduce the computational effort involved in solving such problems.

2. THE STRUCTURE OF THE SOLUTION

Consider a set of g routes (sequences of x_i) given by R_r, $r = 1, \ldots, g$ each one being of the form:

$$R_r = x_0, x_{i_1}, x_{i_2}, \ldots, x_{i_m}, x_0, \tag{1}$$

so that each customer is on exactly one route.

In the above we take R_r to represent both an ordered sequence of customers and also the (unordered) set of these customers.

Considering a route R_r such as given by equation (1) let t_l be the time of arrival of the vehicle (y_{k_r} say) at customer x_{i_l}.

If:

$$B(p, i_l) \geqq t_l \geqq A(p, i_l), \tag{2}$$

for any one period p during which x_{i_l} can accept delivery, then set t'_l, the time of starting the unloading, equal to t_l.

If condition (2) does not apply, then some waiting is involved before the vehicle can begin unloading in which case set:

$$t_l' = \min_p \left[A(p, i_l) \,|\, t_l \leqq A(p, i_l) \right]. \tag{3}$$

If the minimum in expression (3) does not exist then set $t_l' = \infty$.

The time of departure of the vehicle from x_{i_l} is then $t_l' + w_{i_l}$ so that the arrival at the next customer $x_{i_{l+1}}$ is t_{l+1} given by:

$$t_{l+1} = t_l' + w_{i_l} + t(i_l, i_{l+1}). \tag{4}$$

Thus, given a starting time of vehicle y_{k_r} from the depot x_0 as t_0 $(A_{k_r} \leqq t_0 \leqq B_{k_r})$, equations (3) and (4) can be used recursively to calculate t_l for any customer x_{i_l} in the route R_r. (Here we take $w_0 = 0$.) A route R_r can then be feasibly operated by a vehicle y_{k_r} starting at a specfic time d_{k_r} if:

(i) The total working time is satisfied, i. e.:

$$t_{m+1} - d_{k_r} \leqq T_{k_r}, \tag{5}$$

where in calculating t_{m+1} customer $x_{i_{m+1}}$ is taken as the depot x_0. [If during the calculation of the times t_l and t_l' the minimum of expression (3) does not exist and t_l' is set to ∞, route R_r can obviously not be feasibly operated by a vehicle starting at time d_{k_r}.]

(ii) The vehicle y_{k_r} is in the allowable subset Y_i of vehicles of every customer x_i in the route R_r

and

(iii) The sum of the demands q_i of the customers on route R_r is less than the vehicle capacity Q_{k_r}.

2.1. Definition

A set of routes R_r, $r = 1, \ldots, g$ is called feasible if g distinct vehicles y_{k_r}, $r = 1, \ldots, g$ can be found with departure times d_{k_r} so that routes R_r can be operated by the corresponding vehicles y_{k_r} according to conditions (i), (ii) and (iii) above.

2.2. Dominated departure times

From what has already been said earlier, it is quite apparent that the checking of a set of routes for feasibility is not a trivial task, since there is an infinite number of departure times d_{k_r} in the allowable period $B_{k_r} \geqq d_{k_r} \geqq A_{k_r}$. We will now show that all these departure times are totally dominated by a (small) finite number of departure times as far as route feasibility is concerned.

Suppose that a route R_r given by equation (1) is being checked for feasibility when operated by a vehicle y_{k_r} starting at some specific time d_{k_r}.

Suppose that at no stage during the calculation of the arrival times t_l [using equations (3) and (4)], is condition (2) violated. It is then quite apparent that if constraint (5) is satisfied for departure time d_{k_r} it is also satisfied for any other departure time t in the range:

$$d_{k_r} - \min_l [t_l - A(p_l, i_l)] \leqq t \leqq d_{k_r} + \min_l [B(p_l, i_l) - t_l], \tag{6}$$

where p_l refers to the period which satisfies condition (2) for customer x_{i_l}. This is so because within the above range of departure times, condition (2) would never cease to apply and a difference of time Δ in the departure time d_{k_r} would lead to the same difference in time t_{m+1} of constraint (5).

Suppose on the other hand that at some stage l^* during the calculation of arrival times t_l, condition (2) is violated and t'_{l^*} is calculated according to equation (3). In this case it is easily shown that all departure times t in the range:

$$\begin{aligned} d_{k_r} - \min \Big[\min_{l<l^*} \{ t_l - A(p_l, i_l) \}, \{ B((p-1)_{l^*}, i_{l^*}) - t_{l^*} \} \Big] \\ \leqq t \leqq d_{k_r} + \min \Big[\min_{l<l^*} \{ B(p_l, i_l) - t_l \}, \{ t'_{l^*} - t_{l^*} \} \Big], \end{aligned} \tag{7}$$

will leave the value of t'_{l^*} calculated by equation (3) unchanged, and hence as far as constraint (5) is concerned all such departure times are dominated by the RHS of (7). In expression (7) we have used $(p-1)_{l^*}$ to indicate the time period just before period p_{l^*} during which deliveries to customer $x_{i_{l^*}}$ are allowed.

The third possibility during the calculation of the arrival times t_l is when condition (2) is violated but the minimal in (3) does not exist. It is then quite obvious that the route being checked would remain infeasible for all starting times greater than d_{k_r}.

Conditions (6) and (7) then provide means of determining departure periods for vehicles operating a given route. All times within such a period being either indistinguishable as far as route feasibility is concerned [condition (6)], or dominated by a single departure time [condition (7)]. The problem of checking for route feasibility withvarious departure times is then changed from essentially one of infinite number of possibilities to one with only a finite (and in the practical cases small) number of checks.

2.3. Checking a set of routes for feasibility

We have already described how constraint (5) can be used to check a route for time feasibility. The other two conditions [(ii) and (iii) of section 2] can be used directly to determine the subset of vehicles which can feasibly operate

a route. Let an M-row, g-column matrix $E = [e_{ij}]$ be formed with rows corresponding to vehicles and columns corresponding to routes with $e_{ij} = 0$ if vehicle i can feasibly operate route j and $e_{ij} = \infty$ otherwise. The problem of finding a feasible assignment of vehicles to routes is then one of simply solving the assignment problem with the rectangular cost matrix E. [This can be transformed into an ordinary square assignment problem by adding another $(M-g)$ columns with all -0 entries.]

3. METHODS OF SOLUTION

The VRP is quite obviously an extension of the classical travelling salesman problem (TSP) [3, 13, 23]. If the customer requirements C3 and C4 are ignored and we take $Q_k = T_k = \infty$ for the vehicle characteristics V1 and V2, then the VRP with objective O1 becomes the TSP. Even with all the restrictions of the VRP given in section 1.1, the nature of the problem is sufficently close to that of the TSP to allow methods developed for the latter problem to be used in "solving" the VRP. In this section we will discuss methods of "solving" the VRP under three seperate headings:

(i) Exact direct methods.

(ii) Methods based on the TSP

and

(iii) Heuristic methods.

3.1. Exact direct methods

The available algorithms which guarantee optimal solutions to VRP's can only deal with the smallest size problems (*see* section 4). However, the most important of these algorithms are mentioned here because of their theoretical interest and because in certain industries the resulting VRP's are of a small enough mathematical size (although very substantial sums of money may be involved), to be amenable to solution by some of the exact methods. It should also be pointed out here, that all the methods discussed in this section would, if not taken to completion, produce approximate algorithms which may compete with the approximate algorithms described in later sections in both the quality of the answers produced and the corresponding computing times.

A. *Algorithms based on single route enumeration* [2, 13, 25]

These algorithms are really only suitable for VRP's in which all vehicles-in the available set Y of vehicles-are indistinguishable from each other, i. e. Q_k, T_k, B_k and A_k are all constants independent of k and $Y_i = Y$ for all customers $x_i \in X$. It is, however, possible to extend the algorithms to the more

general VRP's of section 1.1 albeit at the cost of even greater computational complexity.

All these methods start by assuming that the totality of routes which a single vehicle can operate feasibly can be generated. Thus, if $S \subseteq X$ is a subset of the customers which can be supplied feasibly on a single route by a vehicle, then it is assumed that the total variable cost associated with the optimal way of routing the customers in S can be calculated. Since the problem of routing optimally the customers in S is a TSP (with time constraints on the arrival times at various customers), this is not a trivial task if $|S|$ happens to be large. However, if these TSP's were to be solved by a branch and bound method such as the sequential version of the algorithm of Little *et al.* [23] rather than one of the methods of Bellmore and Malone [3], Christofides [8] or Held and Karp [17, 18]—which are more suitable for unconstrained TSP's—then the delivery time restrictions help rather than hinder the computations, even allowing for the fact that various departure times have to be checked as mentioned in section [2.2].

Once the feasible single routes for all $S_j \subseteq X$ are calculated, two direct methods of solving the VRP suggest themselves:

Set partitioning algorithm [2, 25]

Set up a matrix $G = [g_{ij}]$ with row i corresponding to customer x_i and a column j for the feasible route through each S_j. Let $g_{ij} = 1$ or 0 depending on whether customer x_i is an element of S_j or not respectively, and let $C(S_j)$ be the cost of the TSP solution through the set of customers in S_j (including the depot), and be associated with column j. The VRP with objective O1 then becomes the problem of choosing a subset of columns with minimum total sum of associated costs, so that each row has an entry of 1 under exactly one of the columns in the chosen subset. This is the well-known set partitioning problem and can be solved optimally for sizes involving several thousand columns. [7, 24] Even so, it should be noted that for a problem with N customers and an average of h customers per route, the number of columns that would need to be generated is of the order of:

$$O\left[\sum_{i=1}^{h}\binom{N}{i}\right],$$

which for a small problem involving, say, $N = 20$ and $h = 5$, becomes over 21,000 columns.

With this formulation, the VRP with objective O2 can be solved in an exactly analogous manner by taking the cost associated with each column to be unity.

The extension of this formulation to the more general case where a vehicle fleet is composed of different vehicles is immediate. In the general case the columns of the matrix $[g_{ij}]$ are feasible route-vehicle pairs so that the total

number of columns could now be up to M times as large as for the simpler case. We would also now have to introduce M additional rows, one for each vehicle, with 1's under the columns in which the vehicle appears as part of the route-vehicle pair and 0's elsewhere. In these additional rows we would now insist that there should be *at most* one entry of 1 in the subset of columns chosen for the solution, since a vehicle can at most be used for one route.

Dynamic programming algorithm [13]

A dynamic programming algorithm could also be used to solve the VRP once all the feasible single routes have been enumerated. In fact this method could be considered as no more than a particular algorithm of solving the set partitioning problem mentioned earlier and will not be discussed here further. The interested reader is referred to [13].

B. *Direct tree search algorithm*

In addition to the algorithms described above where all feasible single routes must be generated *a priori*, a depth-first tree search could be employed in which feasible single routes are generated as and when required. Thus, let us define a node of the search tree to correspond to a feasible single route S_j. The state of the search at stage h can then be represented by an ordered list:

$$L = \{ S_{j_1}(x_{i_1}),\ S_{j_2}(x_{i_2}),\ \ldots,\ S_{j_h}(x_{i_h}) \},$$

where $S_{j_r}(x_{i_r})$ is a feasible single route (the j_r-th) which includes a specified customer x_{i_r} and other customers which are not already included in the previous routes $S_{j_1}(x_{i_1}), \ldots, S_{j_{r-1}}(x_{i_{r-1}})$. The state represented by list L is shown diagrammatically in figure 1, where:

$$F_h = X - \bigcup_{r=1}^{h} S_{j_r}(x_{i_r}),$$

is used for the set of "free" (i. e. as yet unrounted) customers following stage h.

Once the bottom of the tree is reached, say at stage m when $F_m = \varnothing$, the list L contains a solution to the VRP consisting of m routes. Backtracking must then occur in order to consider the possibilities not yet considered. A backtracking step at some general stage h, involves the removal of the last set from L [i. e. the removal of $S_{j_h}(x_{i_h})$] and its replacement by another (as yet unconsidered) feasible single route (say the j'_h-th) passing through customer x_{i_h} to form the new state $L = \{ S_{j_1}(x_{i_1}), \ldots, S_{j'_h}(x_{i_h}) \}$. If no feasible single route through x_{i_h} remains unconsidered, the last-but-one set in L is replaced to form $L = \{ S_{j_1}(x_{i_1}), \ldots, S_{j'_{h-1}}(x_{i_{h-1}}) \}$, etc. Forward branching is then continued from the new state.

A forward branching from some stage h involves the choice of a customer $x_{i_{h+1}} \in F_h$ and the generation of a list $P(x_{i_{h+1}})$ of all feasible single routes

passing through this customer. It is quite apparent that the smaller the number of branching possibilities at any stage h the more efficient the tree search, and it is therefore obvious that $x_{i_{h+1}}$ should be chosen from F_h so as to make the list $P(x_{i_{h+1}})$ as short as possible. This would tend to be the case if $x_{i_{h+1}}$ is chosen to be an "isolated" customer far from the depot. Moreover, not all

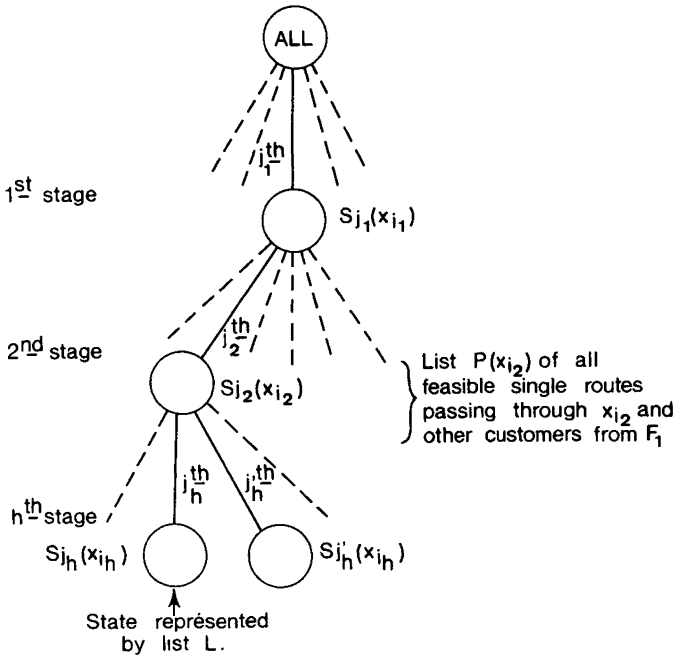

Figure 1.
Direct tree-search algorithm.

nodes produced by $P(x_{i_{h+1}})$ need be kept, and in general, it may be possible to show that a specific node represented by $S_{j^{\alpha}_{h+1}}(x_{i_{h+1}}) \in P(x_{i_{h+1}})$ can be rejected. This could be done in the following circumstances.

(i) If the set of routes already in L together with route $S_{j^{\alpha}_{h+1}}(x_{i_{h+1}})$ fails the joint feasibility test of section 2.3.

(ii) Let F^{α}_{h+1} be the set of free customers implied by the routes in list L and route $S_{j^{\alpha}_{h+1}}(x_{i_{h+1}})$ and let $LB(F^{\alpha}_{h+1})$ be a lower bound on the total variable cost needed to supply F^{α}_{h+1}. Then, if the total cost of the routes in L plus the cost of $S_{j^{\alpha}_{h+1}}(x_{i_{h+1}})$ plus $LB(F^{\alpha}_{h+1})$ is greater than the best answer so far.

(iii) If it could be shown that the remaining free vehicles i. e. those not used for the routes in L or for route $S_{j^{\alpha}_{h+1}}(x_{i_{h+1}})$, are not capable of supplying the customers in F^{α}_{h+1} (e. g. because of insufficient capacity).

(iv) Let $UB(F^{\alpha}_{h+1})$ be an upper bound on the total variable cost needed to supply the customers in F^{α}_{h+1}. Let $S_{j^{\beta}_{h+1}}(x_{i_{h+1}}) \in P(x_{i_{h+1}})$ be another node of the tree produced by $P(x_{i_{h+1}})$. If

$$C[S_{j^{\beta}_{h+1}}(x_{i_{h+1}})] + UB(F^{\beta}_{h+1}) \leqq C[S_{j^{\alpha}_{h+1}}(x_{i_{h+1}})] + LB(F^{\alpha}_{h+1}), \qquad (8)$$

where $C(S)$ is the cost of the single feasible route S, then node $S_{j^{\beta}_{h+1}}(x_{i_{h+1}})$ dominates node $S_{j^{\alpha}_{h+1}}(x_{i_{h+1}})$ and the latter can be removed from the list $P(x_{i_{h+1}})$.

In the above tests it was assumed that upper and lower bounds on the remaining subproblem defined by the free customers could be calculated at any stage. Upper bounds could be calculated by "solving" this subproblem by one of the heuristic methods described in the next section. This provides, in addition to the bound, a possibility of improving the best solution obtained so far, and also other information which could aid the forward branching. Lower bounds could be calculated by relaxing some or all of the constraints and either solving the problem optimally or calculating a lower bound to the solution of the relaxed problem. Any of the available lower bounds for the TSP could be used in this respect [8, 13, 17, 19].

The tree search above has been described for a VRP with objective O1. The applicability of the method to VRP's with objective O2 is obvious and will not be discussed here further.

3.2. Algorithms based on the TSP

If the depot x_0 is replaced by m "artificial" depots $x^1_0, x^2_0, \ldots, x^m_0$, all at the same location as x_0, then a system of m routes operated by m vehicles starting from x_0 could be considered as one single tour starting from x^1_0 visiting some of the customers and returning to x^2_0, visiting some more of the customers and returning to x^3_0, and so on until the tour finally arrives back at x^1_0. The m-route problem is thus converted into a single-tour TSP through $N+m$ points and with restrictions on each section of the TSP tour between any two artificial depots imposed by the route constraints of the VRP. This transformation is achieved quite simply by modifying the inter-customer variable cost matrix $[c(i, j)]$ as shown below [6].

Blocks of m rows and m colums are added as shown with the top left hand submatrix having all entries of λ, all entries in the j-th column of the top right hand submatrix being equal to $c(0, j)$, all entries in the i-th row of the bottom left hand submatrix being equal to $c(i, 0)$, and the bottom right hand submatrix being the initial cost matrix $[c(i, j)]$.

Case (i) ($\lambda = \infty$)

If λ is set to ∞ (i. e. direct travel between the artificial depots is prohibited) then the TSP solution under the cost matrix of figure 2 and with the restrictions

mentioned above will (if feasible) contain exactly m individual routes starting and finishing at a depot.

Case (ii) ($\lambda = -\infty$)

If λ is set to $-\infty$ (and provided m is any number greater than the smallest possible number of vehicles for which the VRP has a feasible solution), then the TSP solution will contain as many inter-depot direct trips as possible thus "shorting out" routes until the minimum number of routes is used. The TSP solution in this case is then the solution to the VRP with objective O2.

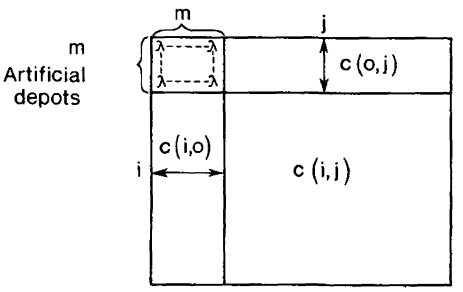

Figure 2.
The modified matrix.

In particular, if λ is a very large-but not infinite-negative number, then the TSP solution will first minimise the number of routes (vehicles) in the solution and secondly find the least variable cost routes using this number of vehicles.

Case (iii) ($\lambda = 0$)

If λ is set to 0 so that there is neither a gain nor a penalty in removing a vehicle, the TSP solution will be the solution to the VRP with objective O1.

Case (iv) ($\lambda = -\varepsilon$)

If in financial terms, the fixed cost of a vehicle is equivalent to ε fewer miles being travelled per day (or per k-day period if the customers are to be supplied once every k-days), then setting $\lambda = -\varepsilon$ produces a TSP solution which minimises the total cost (both fixed and variable) in the VRP.

Because of the restrictions on each section of the TSP tour between artificial depots (imposed by the constraints of the VRP), the solution of such a TSP is very much more difficult than that of an equivalent size pure TSP. As a result, this method of solving VRP's exactly-by using the known exact algorithms for the TSP-is invariably worse than the direct methods described in section 3.1. However, one such exact method based on the TSP is given in [6].

The importance of transforming a VRP into a TSP with restrictions is due to the fact that efficient (both computationally and in terms of the quality of answer) approximate algorithms exist for the latter problem. Particularly useful in this respect are the local optimisation procedures developed by Lin [21], Lin and Kernighan [22] and Christofides and Eilon [9], known as r-optimal methods. These methods start with an initial TSP tour and continuously improve it by taking out r links at a time and replacing them by another set of r links with smaller total cost so as to form an improved tour. In the present context this would imply starting with a set of *m feasible* routes to the VRP (obtained either at random or by the use of a very simple heuristic) and checking (*see* section 2.3) before each exchange of r links, if the resulting tour is composed of a feasible set of actual routes.

3.3. Heuristic methods

There are innumerable heuristic methods both published and unpublished for solving VRP's with various degrees of success. All these methods have as their common base a scoring system which determines which link is to be made or which customer is to be inserted into the routes being constructed. These methods fall into two main categories:

(i) Link-scoring systems

and

(ii) Point-scoring systems.

A. *Methods based on link scoring*

The best known of these heuristics is the "savings" method of Clarke and Wright [11]. Suppose that two customers x_{i_1} and x_{i_2} are supplied individually by two vehicles one visiting x_{i_1} only and the other visiting x_{i_2} only. The total variable cost involved in supplying these customers is

$$c(0, i_1)+c(i_1, 0)+c(0, i_2)+c(i_2, 0).$$

If the customers could be supplied on a single feasible route such as $(x_0, x_{i_1}, x_{i_2}, x_0)$ using only one vehicle, the total variable cost involved would be $c(0, i_1)+c(i_1, i_2)+c(i_2, 0)$ which is a saving of:

$$s_{i_1 i_2} = c(i_1, 0)+c(0, i_2)-c(i_1, i_2). \tag{12}$$

The method initially proposed by Clarke and Wright is to calculate the saving $s_{i_1 i_2}$ for every possible customer pair (x_{i_1}, x_{i_2}) and order these pairs in a list in descending order of savings. Starting from the top of the list one then picks links in sequence which when added to the links previously made form a "feasible" set of routes. A link which violates feasibility is rejected

and the next link in the list is considered. One should note that each link added joins two individual routes into one with a saving in mileage given by equation (12) and a saving of one vehicle. Thus, links are added from the list until no further link can be added feasibly in which case the existing routes are taken to be the solution to the VRP.

The method has a number of shortcomings, the most impotant of which is the fact that the initial "notional" set of routes (where one vehicle visits one customer) is infeasible in most cases since usually $M < N$. This implies that at no stage during the addition of links can total final route feasibility be guaranteed and only individually can routes be checked for feasibility. Thus, for a tightly constrained problem the method can fail to produce a feasible answer. Another obvious shortcoming is that the method does not distinguish between objectives O1 and O2 and that a single answer is obtained.

The first the above-mentioned shortcomings is partly removed by a "one-route-at-a-time" version of the method. In this version the link picked is the topmost in the list of savings amongst those links which join an "end" cutomer of a route being formed with a single customer. This method continuously extends a route until no further link can be added in which case the route is placed in a list of finished routes and another route is started. In this way one can check that the partially completed list of finished routes is at least feasible as a set as explained in section 2.3. The "one-route-at-a-time" version of the savings method was refined considerably (from the computational viewpoint) by Yellow [27] who gives a method that does not require the complete savings list to be calculated, but which can compute efficiently-at each step-the largest-saving link emanating from the "end" customer of the route being formed.

Link scoring measures different from the savings were investigated by Gaskell [14], and other composite measures are used by some of the available commercial codes for solving VRP's [19]. Shightly different ways of using the savings are described by Tillman and Cochran [26] and Knowles [20]. The Clarke and Wright algorithm has been extended by Altman *et al.* [1] to deal with cases where more than one depot is involved and by Beltrami *et al.* [4] to deal with cases where a customer may be visited more than once in a given set of routes.

B. *Methods based on point scoring*

In these methods a score is computed for an unrouted custome to express the "desirability" of it being included into a route being formed. Just as for the methods based on link scoring there are two versions of every point scoring method depending on whether all (say m routes are being formed simultaneously or whether routes are formed "one-at-a-time".

One of the first point scoring methods was given by Hayes [16] who produced a technique which "simulates" the general method which is often used in manual routing. The "one-at-a-time" version of this method proceeds approximately as follows:

A customer is chosen as a "seed" to start a route. A score is computed for every unrouted customer so that: the higher the customer requirement, the distance from the depot and the distance from the nearest other unrouted customer the higher the score, whereas the higher the distance from the straight line joining the "seed" and the depot the lower the score. The customer with the largest score is the one chosen for the next addition into the route being formed (if this is feasible) and the procedure is continued until the route is completed. (Note that some parts of the score may need updating after each addition to the route.) There are several ways that the chosen customer could be added to the route being formed. It could be added at the end of the route, inserted in the best position which causes the least additional mileage to be travelled, or a heuristic method could be used to solve the TSP defined by the customers on the route and the new customer. Another "seed" is now chosen from the remaining unrouted customers to start a new route and so on until all customers are routed. After each completion of a route the subset of the routes that have so far been formed must be checked for feasibility according to section 2.3. The method could be used to generate a number of solutions in different runs by introducing a random element into the score. The best set of routes produced by the various runs would then be chosen.

A different point scoring method was described in a very recent paper by Gillett and Miller [15]. In this method a "seed" is again chosen at random to start a route and a ray (initially joining the depot with the "seed") starts sweeping (clockwise or anticlockwise) with the depot as the pivot. Customers are then added sequentially into the route according to the order in which they are sweeped, (the angle between customer, depot and seed is the score in this case), until the route is completed. The next customer which cannot be feasibly added to the route is now chosen as the seed of the next route to be formed and so on. After all routes are completed the authors suggest customer exchanges between neighbouring routes to reduce the total cost further.

Once more, the point scoring methods do not distinguish between objectives O1 and O2, and although the method of [16] could be particularised for the two cases, the computational performance of the method with objective O2 deteriorates greatly.

4. COMPUTATIONAL COMMENTS AND CONCLUSION

The exact methods discussed in this paper are only suitable for small size problems; the largest VRP that was solved exactly by the present author

La méthode d'optimisation non linéaire sous-jacente (résolution de PP) est la méthode GRG (Abadie et Carpentier [3]), classée première dans le classement de Colville (Colville, [8]). On pourra en trouver un exposé, par exemple, dans le livre de Himmelblau [12].

Dans le tableau qui suit, on désigne par DAK i ou BBB i respectivement l'algorithme DAK ou BBB avec le critère $i = 1, 2, 3, 4$. Le nombre S_k (le « score ») est la valeur de

$$S_k = \frac{1}{7} \sum_{j=1}^{7} \frac{t_{kj}}{\min_l (t_{lj})},$$

où t_{kj} est le temps nécessaire pour résoudre le problème j avec la méthode k. La méthode est d'autant meilleure que S_k est plus petit, l'idéal étant 1. En gros, le rapport des temps de calcul pour deux méthodes k et k' ne devrait pas être trop éloigné de $S_k/S_{k'}$ (Abadie et Guigou [4]) :

Méthode	S	Méthode	S
BBB 1	1.16	DAK 3	1.85
BBB 3	1.18	DAK 1	3.17
BBB 4	1.36	DAK 2	3.22
BBB 2	1.37	DAK 4	3.67

CONCLUSION

Il semble résulter du tableau précédent que BBB soit supérieur à DAK en ce qui concerne la vitesse d'exécution, ce qui n'était nullement évident *a priori*. BBB pourrait donner aussi, comme on l'a vu, une meilleure chance de réussite dans le cas d'un problème non convexe. Enfin, elle possède la qualité que le nombre de mémoires nécessaires à la mémorisation de l'arborescence peut être borné *a priori*.

Si l'on avait à choisir entre les critères, il faudrait peut-être préférer 1 et 3 pour BBB, et 3 en ce qui concerne DAK.

L'expérience faite est encourageante, en ce sens que les problèmes se résolvent assez vite (86 secondes pour les 56 essais sur une machine CDC 6600 sous SCOPE 3 *b*), et que certaines différences paraissent assez significatives. Ce travail devrait être poursuivi, pour atteindre des conclusions plus solides, avec un plus grand nombre de problèmes-test, d'autres critères, plus fins, et peut-être la seconde des deux méthodes BBB mentionnées dans la référence [1].

BIBLIOGRAPHIE

1. J. ABADIE, *Une méthode arborescente pour les programmes partiellement discrets*, R.I.R.O., 3e année, V 3, 1969, p. 24-50.
2. J. ABADIE, *Une méthode de résolution des programmes non linéaires partiellement discrets sans hypothèse de convexité*, R.I.R.O., 5e année, V 1, 1971, p. 23-38.

octobre 1976.

3. J. ABADIE et J. CARPENTIER, *Generalization of the Wolfe Reduced Gradient Method to the Case of Nonlinear Constraints*, in Optimization, R. Fletcher, ed., Academic Press, New York, 1969.
4. J. ABADIE et J. GUIGOU, *Numerical Experiments with the GRG Method*, in Integer and Nonlinear Programming, J. Abadie, ed., North-Holland Publishing Company, Amsterdam, 1970.
5. E. M. L. BEALE et R. E. SMALL, *Mixed Integer Programming by a Branch and Bound Technique*, in Proceedings of the IFIP Congress 1965, p. 450-451, W. A. Kalenich, ed., Spartan Press, Washington D. C., 1965.
6. J. BRACKEN et G. P. MCCORMICK, *Selected Applications of Nonlinear Programming*, Wiley, New York, 1968.
7. A. R. COLVILLE, *A Comparative Study of Nonlinear Programming Codes*, IBM NYSC Report 320-2949, 1968.
8. A. R. COLVILLE, *A Comparative Study of Nonlinear Programming Codes*, p. 487-502, in Proceedings of the Princeton Symposium on Mathematical Programming H. W. Kuhn, ed., Princeton University Press, 1970.
9. R. J. DAKIN, *A Tree-Search Algorithm for Mixed Integer Problems*, The Computer Journal, vol. 8, 1965, p. 250-255.
10. N. J. DRIEBECK, *An Algorithm for the Solution of Mixed Integer Programming Problems*, Management Science, vol. 12, 1966, p. 576-587.
11. P. L. HAMMER et S. RUDEANU, *Méthodes booléennes en recherche opérationnelle*, Dunod, Paris, 1970.
12. D. M. HIMMELBLAU, *Applied Nonlinear Programming*, McGraw-Hill, New York, 1972.
13. A. H. LAND et A. G. DOIG, *An Automatic Method of Solving Discrete Programming Problems*, Econometrica, vol. 28, 1960, p. 497-520.
14. P. LE FAOU, *Une méthode arborescente pour la résolution des programmes linéaires partiellement en nombres entiers*, Thèse de 3e cycle, Université Paris VI, Paris, 1973.
15. J. D. C. LITTLE, K. C. MURTY, D. W. SWEENEY et C. KAREL, *An Algorithm for the Traveling Salesman Problem*, Operations Research, vol. 11, 1963, p. 972-989.
16. J. C. T. MAO, *Quantitative Analysis of Financial Decisions*, The MacMillan Company, Collier-MacMillan Limited, Londres, 1969.
17. J. B. ROSEN et S. SUZUKI, *Construction of Nonlinear Programming Test Problems*, Commun. A.C.M., vol. 8, 1965, p. 113.
18. B. ROY, R. BENAYOUN et J. TERGNY, *From S.E.P. Procedure to the Mixed OPHELIE Program*, in Integer and Nonlinear Programming, J. Abadie, ed., North-Holland Publishing Company, Amsterdam, 1970.
19. J. A. TOMLIN, *Branch and Bound Methods for Integer and Non-Convex Programming*, in Integer and Nonlinear Programming, J. Abadie, ed., North-Holland Publishing Company, Amsterdam, 1970.