J. Opl Res. Soc. Vol. 43, No. 5, pp. 469–481
Printed in Great Britain. All rights reserved

A Branch and Bound Algorithm for a Class of Asymmetrical Vehicle Routeing Problems

GILBERT LAPORTE[1], HÉLÈNE MERCURE[1] and YVES NOBERT[2]
[1]Université de Montréal, Canada, and [2]Université du Québec à Montréal, Canada

This paper describes a branch and bound algorithm for a general class of asymmetrical vehicle routeing problems. Vehicle routes start and end at a central depot. Visits are made to nodes grouped into clusters: every cluster must receive a minimum number of visits. But not all nodes must be visited: there are specified nodes and non-specified nodes. Vehicle routes are also constrained by capacity and distance restrictions. The problem is formulated as an integer linear program. It is then solved by a branch and bound algorithm which exploits the unimodular structure of the subproblems. Computational results are reported.

Key words: vehicle routeing, integer programming

INTRODUCTION

The vehicle routeing problem (VRP) has received considerable attention in Operational Research literature. The most widely studied case of the problem is that which consists of establishing optimal delivery routes from a central depot to a set of geographically scattered points.

Formally, the problem can be described as follows. Consider a graph $G = (N, A, \Gamma, C)$ where

(i) $N = \{1, \ldots, n\}$ is a set of nodes representing cities or customers, node 1 is the depot;
(ii) $A = N^2$ is a set of arcs; note that arcs (i, i) are defined;
(iii) $\Gamma = (d_1, \ldots, d_n)$ is a vector of non-negative weights associated with the nodes; these weights generally represent quantities of goods to be delivered to, or collected from, the nodes; it is assumed that $d_1 = 0$;
(iv) $C = (c_{ij})$ is a matrix of distances or costs associated with the arcs; it is assumed in this paper that C is asymmetrical (if C is symmetrical, our results still hold, but the algorithm developed is not as efficient). C is said to satisfy the triangle inequality if, and only if, $c_{ij} \leqslant c_{ik} + c_{kj}$ for all $i, j, k \in N$.

Based at the depot is a fleet of m identical vehicles of capacity D. One can impose bounds on m: $\underline{m} \leqslant m \leqslant \bar{m}$. If $\underline{m} = \bar{m}$, m is fixed; if $\underline{m} = 1$ and $\bar{m} = n - 1$, m is free. When m is not fixed, it is considered as a decision variable. In this case, we can introduce a non-negative variable cost g on the use of a vehicle. Over a given planning period, some nodes will require the visit of a vehicle (specified nodes) whereas others may not. Moreover, N may be seen as the union of r clusters $S_1, \ldots, S_r$ which need not be disjoint. Assume $S_1 = \{1\}$. It is required to visit at least ν_k nodes in cluster $k (k = 1, \ldots, r)$. Let $T_k \subset S_k$ be the set of specified nodes in S_k; then, without loss of generality, we can set the values of ν_k so that $\nu_k \geqslant |T_k|$. Also, assume that $\nu_1 = 1$ and $T_1 = S_1 = \{1\}$. Let $M = T_1 \cup \ldots \cup T_r$. Also, visiting node i generates a non-negative fixed cost f_i. The VRP considered in this paper can now be stated: determine a set of m vehicle routes in such a way that

(i) all vehicle routes start and end at the depot;
(ii) every specified node of $N - \{1\}$ is visited exactly once by exactly one vehicle;
(iii) at least (or exactly) ν_k nodes per cluster are visited;
(iv) the length of any vehicle route does not exceed a prespecified distance T;

Correspondence: G. Laporte, Centre de recherche sur les transports, Université de Montréal, C.P. 6128, Succursale A, Montréal, Québec, Canada H3C 3J7.

(v) the sum of weights of all specified nodes of any route does not exceed D;
(vi) a convex linear combination of travel cost, node visiting cost and vehicle cost is minimized.

This problem includes, as a special case, the travelling salesman problem (TSP)[1] which consists of determining the shortest Hamiltonian circuit in G. The TSP is obtained by setting $m = 1$, by specifying that all nodes must be visited and by setting $T = \infty$ and $D = \infty$. The problem also contains four well-known types of VRPs, corresponding to constraints (ii), (iii), (iv) and (v) respectively. These are

- the travelling salesman problem with specified nodes—STSP[2];
- the generalized travelling salesman problem through many clusters—GTSP[3];
- the distance constrained vehicle routeing problem—DVRP[4]
- the capacity constrained vehicle routeing problem—CVRP[5].

Comprehensive surveys on these and other types of VRPs can be found in Bodin *et al.*[6], Christofides[7], Laporte and Nobert[8], and Golden and Assad[9].

In addition, the general problem considered in this paper encompasses a large class of real-life vehicle routeing problems combining several characteristics. This will be illustrated by means of a generic example. Consider a two-echelon distribution system with the following components:
(i) a central depot (node 1) used as a base for a fleet of identical vehicles;
(ii) a set of potential sites for the location of secondary facilities; these facilities are visited by vehicles on multiple stop routes; the cost of operating facility i is equal to f_i and d_i units of goods must be picked up from that facility;
(iii) a set of users representing individuals or groups of individuals; over a given planning period, each user makes a number of return trips to the nearest open facility.

The problem consists of simultaneously
(i) selecting an appropriate set of facilities;
(ii) allocating users to facilities;
(iii) establishing a set of feasible vehicle routes through the facilities in order to minimize a function of routeing-operating costs and of user inconvenience.

This example belongs to the general class of network optimization problems described by Golden *et al.*[10]; it can also be viewed as a path location problem[11], as a location-routeing problem[12], and as a location–allocation problem[13].

Labbé and Laporte[11] discuss the case where the depot is a mail sorting office, the facilities are post boxes and the users are groups of people living in the same city block or on the same stretch of road. Here, some components of the problem are left out: only one vehicle is used and there are no capacity or distance restrictions. A second application is the case where the depot is a glass recycling plant and the facilities are glass collection containers or bins posted at various locations in a city. In both examples, the facilities must be located so as to minimize user inconvenience as well as routeing and operating costs. In the post box example, the main constraint imposed on vehicle routes is the total time or distance travelled by the vehicle; in the reusable glass example, vehicle routes are more likely to be constrained by the total amount of glass that can fit into the vehicle. Labbé and Laporte[14] describe a number of modelling approaches for the post box problem. In particular, they show how user travel costs to their nearest open facility can be incorporated into that facility's operating cost.

In this paper, we develop a general algorithm for the VRP, based on the work of Carpaneto and Toth[1] for the TSP. This algorithm incorporates earlier results developed independently for the STSP, the GTSP, the DVRP and the CVRP having an asymmetrical distance structure. Also, additional specifications such as imposing more than one visit per cluster are now considered. By developing a unified algorithm for these problems, we exploited their structural similarities. The advantage of possessing an integrated tool is obvious as it enables the solution of practical problems presenting several of the characteristics described above.

FORMULATION

In this formulation, let x_{ij} be a 0–1 variable defined as follows:
(i) if $i \neq j$, $x_{ij} = 1$ if and only if arc (i, j) appears in the optimal solution;

(ii) if $i = j$, $x_{ij} = 0$ if and only if node i is visited in the optimal solution. (In the following formulation, x_{ii} must be interpreted as 0 whenever $i \in M$.)

Let $\lambda_1, \lambda_2, \lambda_3 \in [0, 1]$ be three preset constants such that $\lambda_1 + \lambda_2 + \lambda_3 = 1$. These reflect the relative weights attached to travel cost, node visiting cost and vehicle cost respectively in the objective function.

The problem is then:

$$\text{minimize} \quad \lambda_1 \sum_{i \neq j} c_{ij} x_{ij} + \lambda_2 \sum_{i \in N} f_i (1 - x_{ii}) + \lambda_3 g m \tag{P}$$

$$\sum_{i=2}^{n} x_{i1} = m \tag{1}$$

$$\sum_{j=2}^{n} x_{1j} = m \tag{2}$$

$$\sum_{\substack{i \neq j \\ i \in N}} x_{ij} = 1 \qquad (j \in M - \{1\}) \tag{3}$$

$$\sum_{\substack{j \neq i \\ j \in N}} x_{ij} = 1 \qquad (i \in M - \{1\}) \tag{4}$$

$$\sum_{i \in N} x_{ij} = 1 \qquad (j \in N - M) \tag{5}$$

$$\sum_{j \in N} x_{ij} = 1 \qquad (i \in N - M) \tag{6}$$

$$\sum_{i \in S_k - T_k} x_{ii} \leqslant (\text{or } =) \; |S_k| - \nu_k \qquad (k = 1, \ldots, r) \tag{7}$$

$$\sum_{\substack{i \neq j \\ i,j \in S}} x_{ij} \leqslant |S| - 1 \qquad (S \subset N \text{ and the nodes of } S \text{ cannot feasibly lie on the same vehicle route}) \tag{8}$$

$$x_{ij} = 0, 1 \qquad (i, j \in N) \tag{9}$$

$$\underline{m} \leqslant m \leqslant \bar{m} \tag{10}$$

In the objective function, it is assumed that all costs are scaled so that they correspond to the same planning period. Constraints (1) and (2) specify that m vehicles must enter and leave the depot. Constraints (3) and (4) state that every specified node must be entered and left exactly once by exactly one vehicle. Constraints (5) and (6) define the degree of non-specified nodes: if node i is not visited, x_{ii} takes the value 1 and the sum of variables associated with arcs entering and leaving the node is then equal to 0; otherwise, x_{ii} is equal to 0 in the constraint associated with node i, forcing an arc to enter the node, and another arc to leave it. Constraints (7) ensure that at least (exactly) ν_k nodes are visited in cluster k: since at least (exactly) $\nu_k - |T_k|$ nodes must be visited in $S_k - T_k$, then at most (exactly) $(|S_k| - |T_k|) - (\nu_k - |T_k|) = |S_k| - \nu_k$ nodes will be unvisited. Constraints (8) ensure that all vehicle routes are feasible. They eliminate all routes including nodes of a set S for which (i) $1 \notin S$ or (ii) $\sum_{i \in S} d_i > D$ or (iii) the length of the shortest Hamiltonian circuit through all nodes of S exceeds T. (This information becomes known during the course of the algorithm—see Step 2). Finally, constraints (9) and (10) specify the integrality conditions and bounds on the variables.

In this formulation, variable x_{ij} need not be defined if $d_i + d_j > D$. Also, if $P(i)$ and $P'(i)$ denote the shortest distance from node 1 to node i and from node i to node 1 respectively, then it is not necessary to define x_{ij} if $P(i) + P'(j) + c_{ij} > T$.

Further reductions can be made in the case where C satisfies the triangle inequality. First, the shortest distance computations are simplified since then, $P(i) = c_{1i}$ and $P'(i) = c_{i1}$. Moreover, the following proposition provides some rules for the elimination of some arcs and nodes from the problem.

Proposition 1

Assume the clusters are disjoint and that C satisfies the triangle inequality. Then, there exists an optimal solution in which only ν_k nodes are visited in cluster $k (k = 1, \ldots, r)$.

Proof

Consider a solution in which more than ν_k nodes are visited in a given cluster k. Let l be a visited node belonging to $S_k - T_k$. In the current solution, $x_{ll} = 0$ and $x_{il} = x_{lj} = 1$ for some i and $j \in N - \{1\}$. Then the cost of an alternative solution without node l, i.e. with $x_{ll} = 1$, $x_{il} = x_{lj} = 0$ and $x_{ij} = 1$, is reduced by at least $\lambda_1(c_{il} + c_{lj} - c_{ij}) + \lambda_2 f_l \geqslant 0$. The conclusion follows by repeating the same argument for all k and for all $l \in S_k - T_k$.

The following corollaries, which hold if the clusters are disjoint and if C satisfies the triangle inequality, can be stated without proof.

Corollary 1

Consider a cluster S_k for which $\nu_k \leqslant 1$. Then all arcs (i, j) for which $i, j \in S_k$ can be eliminated.

Corollary 2

Consider a cluster S_k for which $|T_k| = \nu_k$. Then all nodes $i \in S_k - T_k$ can be eliminated.

Corollary 3

Consider two clusters S_k and $S_{k'}$ with $T_k = T_{k'} = \phi$ and $\nu_k = \nu_{k'} = 1$. Let $i \in S_k$ and $j, h \in S_{k'}$. Then arc (i, j) can be eliminated if $c_{ih} + c_{hl} + f_h \leqslant c_{ij} + c_{jl} + f_j$ for all $l \notin S_k \cup S_{k'}$. Similarly, arc (j, i) can be eliminated if $c_{lh} + c_{hi} + f_h \leqslant c_{lj} + c_{ji} + f_j$ for all $l \notin S_k \cup S_{k'}$.

GRAPH EXTENSION

Through an appropriate extension of the graph G, the model just presented can be transformed into an equivalent model which provides a solution consisting of a single tour. This solution can later be transformed into m distinct vehicle routes. This one-vehicle formulation is particularly useful since we will be using, for the solution of the problem, a modification of the Carpeneto and Toth[1] algorithm which has proven to be very efficient for the single TSP.

There exist several ways to operate the required extension of G. We suggest the following transformation of G into $G' = (N', A', \Gamma', C')$. First adjoin $\bar{m} - 1$ artificial depots to N: let $N^* = \{1, n+1, \ldots, n+\bar{m}-1\}$ be the set of depots and $N' = N \cup N^*$. New clusters are defined as follows: $S_{r+v} = T_{r+v} = \{n+v\}$ $(v = 1, \ldots, \bar{m}-1)$ and M is extended to $M' = M \cup N^*$. $A' = (N')^2$. $\Gamma' = (d_1, \ldots, d_{n+\bar{m}-1})$ where $d_i = 0$ for $i \in N^*$. $C^* = (c'_{ij})$ is defined in such a way that the objective function of (P) can be expressed as a linear combination of the x_{ij} with no constant term:

$$c'_{ij} = \begin{cases} \lambda_1 c_{ij} + \lambda_2 f_i & (i,j \in N, i \neq j) \\ \infty & (i,j \in M, i = j) \\ 0 & (i,j \in N - M, i = j) \\ \lambda_1 c_{i1} + \lambda_2 f_i & (i \in N - \{1\}, j \in N' - N) \\ \lambda_1 c_{1j} & (i \in N' - N, j \in N - \{1\}) \\ -\lambda_3 g_1 & (i = 1, j = n + 1) \\ -\lambda_3 g_v & (i = n + v - 1, j = n + v; \\ & \ v = 2, \ldots, \bar{m} - 1) \\ -\lambda_3 g_{\bar{m}} & (i = n + \bar{m} - 1, j = 1) \\ \infty & \text{otherwise} \end{cases} \tag{11}$$

In (11), g_v can be interpreted as the cost of using vehicle v. Since, in (P), at least $\underline{m}$ vehicles must be used and all vehicles have the same cost g, we set $g_1, \ldots, g_{\underline{m}} = -\infty$ and $g_{\underline{m}+1}, \ldots, g_{\bar{m}} = g$. However, the model and the algorithm can easily be transformed to deal with the case where vehicles have different costs and characteristics. Since a single Hamiltonian circuit is sought over G', the algorithm will systematically avoid arcs $(1, n + 1)$ and $(n + v - 1, n + v)$ for $v = 2, \ldots, \underline{m}$, thus forcing a vehicle route between the origin and the end of each of these arcs. With these definitions, the third term of the objective function of (P) can be rewritten as $\lambda_3(\bar{m}g - g_1 x_{1,n+1} - g_{\bar{m}} x_{n+\bar{m}-1,1} - \sum_{v=2}^{\bar{m}-1} g_v x_{n+v-1,n+v})$. The constant term $\lambda_3 \bar{m} g$ can, of course, be removed. The new formulation (P') can be stated as follows:

$$\text{minimize} \quad z = \sum_{i,j \in N'} c'_{ij} x_{ij} \tag{P'}$$

subject to constraints (7),

$$\sum_{\substack{i \neq j \\ i,j \in S}} x_{ij} \leqslant |S| - 1 \qquad (S \subset N', S \cap N \neq \phi \text{ and the nodes of } S \text{ cannot feasibly lie on the same vehicle route}) \tag{8'}$$

and

$$\sum_{i \in N'} x_{ij} = 1 \qquad (j \in N') \tag{12}$$

$$\sum_{j \in N'} x_{ij} = 1 \qquad (i \in N') \tag{13}$$

$$x_{ij} = 0, 1 \qquad (i,j \in N') \tag{14}$$

Note that when (7) and (8′) are relaxed, (P') reduces to an assignment problem (AP). In order to transform a single route solution to (P') into an m-route solution over G, it suffices to apply the following rules. Consider the set of arcs $(i,j) \in A'$ for which $i \neq j$ and $x_{ij} = 1$:
(i) if $i \notin N^*$ and $j \in N^*$, replace (i,j) by $(i,1)$;
(ii) if $i \in N^*$ and $j \notin N^*$, replace (i,j) by $(1,j)$;
(iii) if $i,j \in N^*$, remove (i,j).

Finally, it will prove useful later to know an upper bound $\bar{z}$ on the value of the optimal solution to problem (P'). The maximum cost is obtained when all nodes are visited and $\bar{m}$ vehicles travel the maximum allowable distance. In such a case, no arc (i,j) for which $i,j \in N^*$ is used. The value of $\bar{z}$ can therefore be set equal to

$$\bar{z} = \lambda_1 \bar{m} T + \lambda_2 \sum_{i \in N} f_i \tag{15}$$

ALGORITHM

As mentioned earlier, (P') is solved by applying to the extended graph G', a modified version of the Carpeneto and Toth[1] branch and bound algorithm for the TSP. We have borrowed the main structure of this algorithm but we have modified it in several ways. Here are the main changes:

(i) the subproblems are assignment problems (APs) instead of modified assignment problems, i.e. we allow assignments on the main diagonal, corresponding to setting variables x_{ii} at 1;
(ii) preprocessing allows the elimination of several arcs from the problem;
(iii) during the course of the algorithm, other arcs may be excluded from the solution in order to avoid the generation of subproblems in which it would be impossible to satisfy either the capacity or the maximum route length constraint;
(iv) branching is made on nodes as well as on arcs;
(v) a different procedure for the computation of lower bounds on the value of the optimal solution is introduced.

The following notation will be used.

z^*	the cost of the best feasible solution to (P') so far identified; at the end of the algorithm: the optimum;
$\bar{z}$	an upper bound on z^* (see (15));
z_p	the value of the objective function of the AP at node p of the search tree;
$\underline{z}_p$	a lower bound on z_p;
IA_p	the set of included arcs in subproblem p;
EA_p	the set of excluded arcs in subproblem p;
IN_p	the set of included nodes in subproblem p;
EN_p	the set of excluded nodes in subproblem p.

The algorithm can be broken down into the following steps.

Step 0 (pre-processing). Remove from G all infeasible or dominated arcs or nodes (see Proposition 1). Eliminating an arc is done by setting its cost to infinity. Dominated nodes are simply dropped from the graph. Transform the resulting graph into an extended graph G' according to the rules described above. If a feasible solution to (P') is known, let z^* be the cost of that solution. Otherwise, set $z^* = \infty$.

Step 1 (node 1 of the search tree). Set $IA_1 = EA_1 = EN_1 = \phi$, $IN_1 = M'$ and solve the AP resulting from the relaxation of constraints (7) and (8′) in (P'). If $z_1 = z^*$, terminate.

Step 2 (feasibility check). Check whether the solution is illegal. There are two classes of infeasibilities.

(i) There exists a cluster k having fewer than ν_k visited nodes, i.e. such that $\sum_{i \in S_k} (1 - x_{ii}) < \nu_k$.

(ii) There exists at least one illegal circuit or path in G', i.e. a circuit or a path corresponding to an illegal subtour in G. In order to identify illegal circuits or paths in G', consider the solution corresponding to the current subproblem. It consists of one or several Hamiltonian circuits. Subtours over subsets of $N' - N^*$ are illegal since they are disconnected from the depot in G. Other subtours can be broken down into paths of the form (i_1, i_2) where $i_1, i_2 \in N^*$ or of the form $(i_1, i_2, \ldots, i_t)$ where i_1, $i_t \in N^*$ and $i_2, \ldots, i_{t-1} \in N' - N^*$. Paths (i_1, i_2) pose no problem since they correspond to arcs which will eventually be removed. The other types of paths correspond to vehicle routes. These are illegal if they are either too heavy, i.e. $\sum_{j=2}^{t-1} d_{i_j} > D$ or too long, i.e. $\sum_{j=1}^{t-1} c_{i_j i_{j+1}} > T$. If the solution is feasible, terminate. Otherwise, insert node 1 of the search tree in a queue.

Step 3 (node selection). If the queue is empty, terminate. Otherwise, select the next node (node p) from which to branch and remove it from the queue. Branching is always done on the pending node having the smallest z_p and such that $z_p < z^*$. (If $z_p \geqslant z^*$, the algorithm terminates).

Step 4 (branching on nodes). The solution found at node p is illegal and must be eliminated. If all clusters S_k include at least ν_k visited nodes, proceed to Step 5. Otherwise, consider all clusters S_k having fewer than ν_k visited nodes and for each of them, define:

V_k the set of nodes which are currently visited (note: $T_k \subseteq V_k$); and

R_k the set of unvisited nodes which could eventually become part of the solution:

$$R_k = S_k - V_k - EN_p.$$

- If $|R_k| < \nu_k - |V_k|$ for some k, the current problem is infeasible: proceed to step 3.
- If $|R_k| = \nu_k - |V_k|$ for all k, create a new subproblem q with

$$IN_q = IN_q \cup (\bigcup_k R_k)$$
$$EN_q = EN_p$$

  and skip the remainder of Step 4 as well as Steps 5 and 6.
- Otherwise, let $K = \{k : |R_k| = \nu_k - |V_k|\}$. If $K \neq \phi$, set $IN_q = IN_p \cup (\bigcup_{k \in K} R_k)$. Let k^* be the index for which $\nu_k > |V_k|$, $|R_k| > \nu_k - |V_k|$ and $|R_k|$ is minimized. Label the nodes of R_{k^*} as $i_1, i_2, \ldots, i_s$ (note that $s > 1$). Subproblems q are then created from subproblem p by updating the sets of included and excluded nodes as follows:

$$IN_q = IN_p \cup \{i_q\} \qquad (q = 1, \ldots, s)$$

$$EN_q = \begin{cases} EN_p & (q = 1) \\ EN_p \cup \{i_u : u = 1, \ldots, q-1\} & (q = 2, \ldots, s). \end{cases}$$

In problems in which clusters are disjoint and C satisfies the triangle inequality, and in problems in which exactly ν_{k^*} nodes must be visited in cluster S_{k^*}, it is valid to exclude all nodes of $S_{k^*} - IN_q$ when $|S_{k^*} \cap IN_q| \geqslant \nu_{k^*}$. In this case, IN_q is defined as above and EN_q becomes

$$EN_q = EN_p \cup (S_{k^*} - IN_q) \quad (q = 1, \ldots, s).$$

This is clearly stronger than the rule used in the general case. In practice, including a node i in IN_q is done by forcing variable x_{ii} to 0 (by setting its cost to infinity) in subproblem q; similarly, including a node i in EN_q is done by forcing variable x_{ii} to 1 in subproblem q. Skip Steps 5 and 6.

Step 5 (branching on arcs). All clusters include a sufficient number of visited nodes, but the solution contains illegal subtours or paths. Eliminating them requires removing one of their arcs from the solution. Consider the illegal subtour or path having the least number of arcs not already included in IA_p and let these arcs be $(i_1, j_1), \ldots, (i_a, j_a)$. Here the arcs are listed in the same order in which they appear on the subtour or path. Then define

$$IA_q = \begin{cases} IA_p & (q = 1) \\ IA_p \cup \{(i_u, j_u) : u = 1, \ldots, q-1\} & (q = 2, \ldots, a) \end{cases}$$

$$EA_q^0 = EA_p \cup \{(i_q, j_q)\} \qquad (q = 1, \ldots, a).$$

Step 6 (excluding additional arcs). To EA_q^0, the set of excluded arcs in subproblem q, can be adjoined three other sets K_1, K_2 and K_3 of excluded arcs, defined as follows. In subproblem q, N' can be partitioned into $Q(N') = \{W_1, \ldots, W_b\}$ where $W_1, \ldots, W_b$ are subsets of nodes linked by the arcs of IA_q. Some of these subsets are singletons whereas the others correspond to paths of included arcs.

(i) *Arcs excluded in order to prevent the occurrence of subtours having a total weight exceeding D*

Every subset W_t has a weight $w(W_t) = \sum_{i \in W_t} d_i$. If W_t corresponds to a path, this path has two end nodes i_t' and i_t''. Let $E_t = \{i_t', i_t''\}$. If W_t is a singleton, i_t' and i_t'' coincide. Any subsets W_t and W_u cannot belong to the same route in the optimal solution if their total weight exceeds D. It is therefore valid to exclude arcs linking end nodes of subsets whose total weight exceeds D. The set of these excluded arcs will be designated by $K_1(IA_q)$. Formally,

$$K_1(IA_q) = \{(i,j) \in (N' - N^*)^2 : i \in E_t, j \in E_u,\ W_t,\ W_u \in Q(N'),$$
$$\neq w(W_t) + w(W_u) > D\}.$$

(ii) *Arcs excluded in order to prevent the occurrence of subtours having a total length exceeding T*

Now consider a subset W_t corresponding to a path. W_t has a length equal to $y(W_t) = \sum_{\substack{i,j \in W_t \\ i \neq j}} c_{ij}$, where (i,j) is an arc of W_t. Again, let i_t' and i_t'' be the first and the last node of the path associated with W$_t$. First observe that subproblem q has no feasible descendant if $P(i_t') + P'(i_t'') + y(W_t) > L$. In such a case, subproblem q can be eliminated from consideration in the remainder of the algorithm. Moreover, arc (i_t'', i_t') can be excluded from subproblem q and all its descendants if

(a) $i_t', i_t'' \in N' - N^*$.

Any arc (i, i_t') can be eliminated likewise if

(b) $P(i) + P'(i_t'') + y(W_t) + c_{i,i_t'} > T$.

Similarly, any arc (i_t'', j) can be eliminated if

(c) $P(i_t') + P'(j) + y(W_t) + c_{i_t'',j} > T$.

Now, consider two subsets W_t and W_u corresponding to paths of arcs of IA_q. Suppose $i_t', i_t'', i_u', i_u'' \in N' - N^*$, then:

(d) arc (i_t'', i_u') can be excluded if $P(i_t') + P'(i_u'') + y(W_t) + y(W_u) + c_{i_t'',i_u'} > T$,

(e) similarly, arc (i_u'', i_t') can be excluded if $P(i_u') + P'(i_t'') + y(W_t) + c_{i_u'',i_t'} > T$.

Let $K_2(IA_q)$ be the set of all arcs eliminated through one of the above five criteria.

(iii) *Replicas of excluded arcs between N^* and $N' - N^*$*

Let $EA_q^1 = EA_q^0 \cup K_1(IA_q) \cup K_2(IA_q)$ and

$$K_3(EA_q^1) = \bigcup_{j \in N^*} [\{(i',j) : (i',i'') \in EA_q^1,\ i' \in N' - N^*,\ i'' \in N^*\}$$
$$\cup \{(j,i'') : (i',i'') \in EA_q^1,\ i' \in N^*,\ i'' \in N' - N^*\}].$$

In other words, if an excluded arc has one node in N^* and another node i^* in $N' - N^*$, then all arcs having a node i^* in $N' - N^*$ and another node in N^* are also excluded.
Then set $EA_q = EA_q^1 \cup K_3(EA_q^1)$.

For each remaining subproblem q, execute Steps 7 to 10. Then go to Step 3.

Step 7 (AP bound). Compute a lower bound $\underline{z}_q$ on z_q according to Step 3 of Carpaneto and Toth[1]. If $\underline{z}_q \geqslant z^*$ or $\underline{z}_q > \bar{z}$, consider the next q and repeat Step 7.

Step 8 (AP solution). Solve the AP associated with node q. This problem is constrained by the sets IA_q, EA_q, IN_q and EN_q. If $z_q \geqslant z^*$ or $z_q > \bar{z}$, consider the next q and proceed to Step 7.

Step 9 (feasibility check). Check whether the current solution is feasible. If so, set $z^* = z_q$ and store the tour; if $z^* = z_p$, proceed to Step 3; otherwise, consider the next q and proceed to Step 7. If the solution contains at least one illegal subtour or path, consider all chains of nodes linked by arcs of IA_q. Let M_k be the set of nodes of the kth such chain. Now let x_{ij} be a binary variable equal to 1 if and only if arc (i,j) belongs to the solution. It can be shown[4,5] that for every subset U of N, the number of vehicles required to serve all nodes of U in the optimal solutions is bounded below by $V(U) = \left\lceil \dfrac{\sum_{i \in U} d_i}{d} \right\rceil$ and by $V(U) = \max \left\{ \left\lceil \dfrac{\sum_{i \in U} d_i}{D} \right\rceil, \left\lceil \dfrac{\sum_{i,j \in U} c_{ij}x_{ij}}{T} \right\rceil \right\}$

if C satisfies the triangle inequality. Here, $\lceil t \rceil$ denotes the smallest integer greater than or equal to t if $t > 0$ and is equal to 1 otherwise. It is therefore valid to force at least $V(U)$ links between U and its complement:

$$\sum_{i \in U, j \in N - U} x_{ij} \geqslant V(U) \tag{18}$$

or, equivalently, at most $|U| - V(U)$ arcs within U:

$$\sum_{i,j \in U} x_{ij} \leqslant |U| - V(U) \tag{19}$$

This equivalence holds as long as the degree of every node in U is equal to 2 (see Laporte[15]). Since this condition applies to $U = M_k$, we can impose

$$\sum_{i,j \in M_k} x_{ij} \leqslant |M_k| - V(M_k) \tag{20}$$

But if this condition cannot be met for at least one k, i.e. if $V(M_k) > 1$, subproblem q has no feasible descendant. In such a case, consider the next q and proceed to Step 7.

Step 10 (alternative lower bound). Compute a new lower bound $\underline{z}'_q$ on the value of the optimal TSP solution associated with subproblem q. The validity of the bound rests on the fact that this solution will consist of one Hamiltonian circuit through every node of IN_q and through at least ν_k nodes from each cluster k. The bound is obtained by modifying a procedure developed by Christofides[16] for the TSP:

(i) Remove from N' all nodes belonging to EN_q and extract from C' the distance matrix $\Delta = (d_{ij})$ associated with $(N' - EN_q)^2$.

(ii) Contract every cluster S_k having an empty intersection with IN_q and for which $\nu_k \geqslant 1$ into a single node i: if two clusters S_{k_1} and S_{k_2} are contracted into nodes i_{k_1} and i_{k_2} respectively, then the distance between these two nodes is equal to $d_{i_{k_1} i_{k_2}} = \min_{u \in S_{k_1},\, v \in S_{k_2}} \{d_{uv}\}$. Let L be the set of indices of all contracted clusters.

(iii) Let $N_q = IN_q \cup \{i_k : k \in L\}$ and Δ_q be the distance matrix associated with $(N_q)^2$. Set $\underline{z}'_q = 0$.

(iv) Compress Δ_q. This operation consists of transforming Δ_q into a matrix which satisfies the triangle inequality. This is done by replacing every d_{ij} for which $d_{ij} > d_{ih} + d_{hj}$ for some h by $\min_h \{d_{ih} + d_{hj}\}$. This process is repeated until Δ_q satisfies the triangle inequality.

(v) Solve the AP associated with Δ_q. Let v^* be the value of its optimal solution. Let $\underline{z}'_q = \underline{z}'_q + v^*$.

(vi) Contract the set of nodes associated with every subtour into a single node and compute the associated Δ_q matrix.

(vii) Repeat (iv), (v) and (vi) until Δ_q is a 1×1 matrix.

(viii) $\underline{z}'_q$ is a valid lower bound on the value of the optimal TSP solution on N_q (see Christofides[16] p 1049) and hence, on the value of the VRP solution associated with subproblem q. If $\underline{z}'_q \geqslant z^*$ or $\underline{z}'_q > \bar{z}$, consider the next q and proceed to Step 7. Otherwise, insert node q in the queue.

COMPUTATIONAL RESULTS

The algorithm was tested on a number of problems. Its performance was evaluated and compared for various combinations of the following characteristics:

(i) problem size;
(ii) distance matrix satisfying the triangle inequality or not;
(iii) percentage of specified nodes;
(iv) number of clusters;
(v) distance limit T;
(vi) vehicle capacity D.

With such a general algorithm, it is unrealistic to carry out extensive tests on all possible combinations. Some parameters and characteristics were therefore fixed for all test problems:

(i) all clusters were disjoint and had approximately the same cardinality; apart from S_1 which always contained only one node, specified nodes were randomly distributed among the clusters;

(ii) ν_k was set equal to $\max\{1, |T_k|\}$ for all clusters S_k;

(iii) only distance was considered in the objective function, i.e. we set $\lambda_1 = 1$, $\lambda_2 = \lambda_3 = 0$;

(iv) $\underline{m}$ was defined as $\left\lceil \frac{\sum_{i \in N} d_i}{D} \right\rceil$ and $\bar{m}$ was set equal to $\underline{m} + 2$.

The distance matrices were generated as follows.

Problems in which C satisfied the triangle inequality

Two sets of n points $\{P_1, \ldots, P_n\}$ and $\{Q_1, \ldots, Q_n\}$ were first independently generated in $[0,100]^2$ according to a uniform distribution. Then c_{ij} was set equal to $\|P_i - P_j\|$ for $i < j$, to $\|Q_i - Q_j\|$ for $i > j$ and to 0 for $i = j$. All distances were then rounded up or down to the nearest integer and the C matrix was compressed as in Step 10 of the algorithm.

Problems in which C did not satisfy the triangle inequality

All c_{ij} were randomly generated according to a uniform distribution on [0,100] and then rounded up or down to the nearest integer.

For every value of n considered and for each of the two types of distance matrices, three different problems were independently generated. These three problems were then attempted for various values of the parameters under study.

For the weights d_i and the vehicle capacity D, the following procedure was used in order to produce problems of various degrees of tightness. The d_i were first generated according to a uniform distribution on [0,100] and rounded up and down to the nearest integer. D was then defined as $(1 - \alpha) \max_i \{d_i\} + \alpha \sum_{i \in N} d_i$ where $0 \leqslant \alpha \leqslant 1$: problems become tighter as α becomes smaller.

The value of T was set at different levels β. At level $\beta = 0$, the value of T was equal to ∞, lifting, in fact, the maximum distance restriction. Letting $b(\beta)$ be the length of the longest route in the optimal solution of the problem solved at level β and $T(\beta)$, the corresponding value of T, we defined the subsequent distance limits as $T(\beta) = 0.9 \times b(\beta - 1)$ for $\beta > 1$, again producing tighter and tighter problems.

All problems were run on the University of Montreal CYBER 855 computer, using an FTN5 compiler, but all reported times are CYBER 173 equivalent. A time limit of 300 seconds was imposed for the solution of any problem. 'Successful problems' are those which could be solved to optimality within that time limit.

Computational results are reported in Tables 1 to 3. The meanings of the various column headings are as follows.

n	number of cities (before transformation of G into G');
α	capacity tightness parameter (see above);
β	distance tightness parameter (see above);
γ	percentage of nodes in $N - \{1\}$ which were specified nodes ($\gamma = 100(\lvert M \rvert - 1)/(n - 1)$);
r	number of clusters (before transformation of G into G');
%ARCS	percentage of arcs eliminated in the pre-processing phase (Step 1 of the algorithm);
SUCC	number of successful problems out of 3.

For the following parameters, all reported results represent average values over the successful problems

AP	number of assignment problems solved in the branch and bound algorithm (see Step 8);

QUEUE	number of nodes inserted in the queue (see Steps 2 and 10);
DESC	number of these nodes which were later examined (i.e. which had at least one descendant—see Step 3);
SETS	number of clusters having fewer than ν_k visited nodes requiring branching (see Step 4);
SUBTOURS	number of subtours disconnected from N^* (see Step 5);
DIST	number of paths corresponding to vehicle routes longer than T (see Step 5);
CAPAC	number of paths corresponding to vehicle routes having a total weight exceeding D (see Step 5);
ELIM	number of arcs eliminated at Step 6 of the algorithm;
TIME	number of CPU seconds.

Here are the main conclusions that can be drawn from Tables 1 to 3:

(i) The algorithm was quite successful in optimally solving problems involving up to 40 cities. In 'pure problems' containing only one of the four main characteristics studied, sizes of up to about 100 could be reached[2-5]. But we are dealing here with much harder problems compounding four types of difficulties. To our knowledge, never in the past has an exact algorithm been devised and tested on vehicle routeing problems presenting such a level of complexity and such a variety of constraints.

(ii) When only one parameter at a time is allowed to vary, the difficulty of the problem increases with n, β, γ and r; it decreases with α. This is consistent with results obtained on pure DVRPs[4], CVRPs[5], and GTSPs[3]. As far as the pure STSP is concerned, it has been noted[2] that computation times are not monotonically related to γ: there appears to be a peak at $\gamma = 0.25n$ and a trough at $\gamma = 0.75n$. It has not been possible to observe this behaviour in this study, partly because of the small range chosen for γ, of the small sample size and of the various interactions with other parameters.

(iii) The percentage of eliminated arcs in the pre-processing phase (%ARCS) was, in general, larger in problems for which the C matrix satisfied the triangle inequality. This was of course to be expected. What is new with respect to previous studies is that these problems also tended to be easier. For example, in a paper on the symmetrical GTSP, Laporte and Nobert[17] had observed that, in spite of having a relatively higher percentage of eliminated arcs, problems in which C satisfied the triangle inequality were still more difficult to solve.

(iv) The number of arcs eliminated in Step 6 of the algorithm (ELIM) becomes more important as β becomes large and as α becomes small. But this beneficial effect is annihilated by the number of illegal solutions (see the DIST and CAPAC columns) generated when the problems become more constrained. The net effect is unfortunately negative.

TABLE 1. *Computational results for 30-city problems in which* C *satisfied the triangle inequality*

α	β	γ	r	%ARCS	SUCC	AP	QUEUE	DESC	SETS	SUBTOURS	DIST	CAPAC	ELIM	TIME
0.50	0	0	4	53.9	3	13	5	2	5	0	0	0	0	1.16
			6	29.9	3	52	38	11	33	5	0	0	0	6.43
			12	14.8	3	2466	1184	1014	422	762	0	0	0	117.26
		20	3	89.8	3	1	0	0	0	0	0	0	0	0.06
			5	85.2	3	2	1	1	0	1	0	0	0	0.15
			8	43.9	3	150	86	55	28	55	0	1	0	7.25
		50	7	67.0	3	205	66	62	0	46	0	20	0	5.00
			12	67.0	3	205	66	62	0	46	0	20	0	5.06
			20	58.6	3	1649	411	400	7	383	0	20	0	30.46
0.50	1	0	4	84.9	3	345	152	152	40	106	6	0	0	14.30
			6	76.3	1	562	302	264	50	105	147	0	0	19.40
			12	27.3	0	–	–	–	–	–	–	–	–	–
		20	3	89.8	3	27	16	12	0	6	9	0	1	0.60
			5	83.1	3	182	103	79	4	44	55	0	22	4.65
			8	44.4	1	1698	660	592	145	305	115	95	115	62.45
		50	7	66.9	2	3188	1335	969	0	963	372	0	235	102.91
			12	66.9	2	3188	1335	969	0	963	372	0	235	102.29
			20	58.6	1	2683	963	806	35	544	386	0	52	79.95

TABLE 2. *Computational results for 30-city problems in which* C *did not satisfy the triangle inequality*

α	β	γ	r	%ARCS	SUCC	AP	QUEUE	DESC	SETS	SUBTOURS	DIST	CAPAC	ELIM	TIME
0.50	0	0	4	0.0	3	1	0	0	0	0	0	0	0	0.12
			6	0.0	3	15	13	3	11	2	0	0	0	1.70
			12	0.0	3	97	58	42	49	8	0	0	0	9.51
		20	3	0.0	3	5	1	0	0	2	0	0	0	0.32
			5	0.0	3	5	2	1	0	2	0	0	0	0.32
			8	0.0	3	21	9	6	3	5	0	1	0	1.02
		50	7	0.0	2	1448	788	289	0	460	0	328	108	65.84
			12	0.0	2	1448	788	289	0	460	0	328	108	66.00
			20	0.0	2	800	427	182	22	250	0	154	154	35.10
0.50	1	0	4	84.9	3	100	10	10	10	10	0	0	0	2.24
			6	83.8	3	246	95	95	45	43	6	0	0	10.05
			12	75.5	3	618	332	332	108	53	171	0	3	46.27
		20	3	63.0	1	447	226	123	0	154	51	21	121	28.04
			5	63.0	1	447	221	120	4	145	51	21	120	27.79
			8	57.4	1	290	115	69	21	69	13	15	56	15.50
		50	7	29.8	1	1033	278	250	0	155	3	120	205	44.91
			12	29.8	1	1033	278	250	0	155	3	120	205	44.28
			20	29.8	1	712	222	190	1	153	3	65	151	30.27

TABLE 3. *Computational results for problems in which* $\alpha = 0.60$

			C satisfied the triangle inequality						*C* did not satisfy the triangle inequality					
			$\beta = 1$			$\beta = 2$			$\beta = 1$			$\beta = 2$		
n	γ	r	SUCC	AP	TIME	SUCC	AP	TIME	SUCC	AP	TIME	SUCC	AP	TIME
20	10	3	3	1	0.06	3	4	0.15	3	59	0.99	3	40	0.41
		6	3	30	0.70	3	154	3.85	3	531	14.92	3	486	15.11
	40	3	3	483	7.79	3	344	6.78	3	1093	31.38	3	2955	86.50
		6	3	1855	30.97	2	337	6.27	3	1102	31.02	3	3198	91.52
30	10	3	3	1	0.05	3	76	1.16	1	1	0.08	1	1	0.08
		6	3	3061	88.92	3	48	2.35	1	1	0.08	1	216	4.01
	40	3	3	1048	24.92	2	1522	38.41	1	1914	99.04	0	–	–
		6	3	1048	25.05	2	1522	38.52	1	1914	98.74	0	–	–
40	10	3	3	28	0.84	3	0	0.07	0	–	–	0	–	–
		6	2	1442	64.04	3	2423	100.23	0	–	–	0	–	–
	40	3	1	23	0.98	0	–	–	0	–	–	0	–	–
		6	1	23	0.97	0	–	–	0	–	–	0	–	–

(v) Similarly, as r and β become larger, solutions in which clusters are visited less than ν_k times are more likely to occur, causing more branching (see the SETS column).

CONCLUSION

We have presented in this paper an ILP formulation and an exact algorithm for a variety of single depot vehicle routeing problems possessing one or several characteristics:

- a fixed or variable number of vehicles;
- specified nodes and non-compulsory nodes;
- a minimum number of visits per cluster of nodes;
- distance constraints;
- capacity constraints;
- an objective function incorporating routeing costs, node visiting costs and vehicle costs.

As mentioned earlier, the formulation and the algorithm can easily be modified to deal with a fleet of heterogeneous vehicles. The algorithm developed has been applied to the exact solution of problems involving up to 40 cities. Previous results confirm that a similar approach can be used to solve larger 'pure problems'. To our knowledge, never in the past has an exact algorithm been devised for the solution of routeing problems possessing so many characteristics and such a level of difficulty.

Acknowledgements—The authors are grateful to the Canadian Natural Sciences and Engineering Research Council (grants A4747 and A5486) and to the Quebec Government (FCAR grant 86EQ3084) for their financial support. Thanks are also due to Giorgio Carpaneto and Paolo Toth who made their TSP code available to the authors. Paolo Toth also made a number of valuable comments on a preliminary version of this paper; this help is gratefully acknowledged.

REFERENCES

1. G. Carparneto and P. Toth (1980) Some new branching and bounding criteria for the asymmetrical travelling salesman problem. *Mgmt Sci.* **26**, 736–743.
2. G. Laporte, H. Mercure and Y. Nobert (1984) Optimal tour planning with specified nodes. *RAIRO (recherche opérationnelle)* **18**, 203–210.
3. G. Laporte, H. Mercure and Y. Nobert (1987) Generalized travelling salesman through *n* sets of nodes: the asymmetrical case. *Disc. Appl. Math.* **18**, 283–289.
4. G. Laporte, Y. Nobert and S. Taillefer (1987) A branch-and-bound algorithm for the asymmetrical distance-constrained vehicle routing problem. *Mathematical Modelling* **9**, 857–868.
5. G. Laporte, H. Mercure and Y. Nobert (1986) An exact algorithm for the asymmetrical capacitated vehicle routing problem. *Networks* **16**, 33–46.
6. L. D. Bodin, B. L. Golden, A. A. Assad and M. O. Ball (1983) Routing and scheduling of vehicles and crews. The state of the art. *Comps Opns Res.* **10**, 69–211.
7. N. Christofides (1985) Vehicle routing. In *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization* (E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan and D. B. Shmoys, Eds) pp. 431–448. Wiley, Chichester.
8. G. Laporte and Y. Nobert (1987) Exact algorithms for the vehicle routing problem. In *Surveys in Combinatorial Optimization* (S. Martello, G. Laporte, M. Minoux and C. Ribeiro, Eds) pp. 147–184. North-Holland, Amsterdam.
9. B. L. Golden and A. A. Assad, (Eds) (1988) *Vehicle Routing: Methods and Studies*. North-Holland, Amsterdam.
10. B. L. Golden, M. O. Ball and L. D. Bodin (1981) Current and future research directions in network optimization. *Comps Opns Res.* **8**, 71–81.
11. J. Current, C. S. ReVelle and J. L. Cohon (1985) The maximum covering/shortest path problem: a multiobjective network design and routing formulation. *Eur. J. Opl Res.* **21**, 189–199.
12 G. Laporte (1988) Location-routing problems. In *Vehicle Routing: Methods and Studies* (B. L. Golden and A. A. Assad, Eds) pp. 163–197. North-Holland, Amsterdam.
13. L. Cooper (1963) Location-allocation problems. *Opns Res.* **11**, 331–343.
14. M. Labbé and G. Laporte (1986) Maximizing user convenience and postal service efficiency in post box location. *Belgian J. Opns Res. Stat. Comp. Sci.* **26**, 21–35.
15. G. Laporte (1986) Generalized subtour elimination constraints and connectivity constraints. *J. Opl Res. Soc.* **37**, 509–514.
16. N. Christofides (1972) Bounds for the travelling salesman problem. *Opns Res.* **20**, 1044–1056.
17. G. Laporte and Y. Nobert (1983) Generalized travelling salesman through *n* sets of nodes: an integer programming approach. *INFOR* **21**, 61–75.