Journal of the Operational Research Society (1996) 47, 329–336 0160-5682/96 $12.00

An Improved Petal Heuristic for the Vehicle Routeing Problem

JACQUES RENAUD[1–3], FAYEZ F. BOCTOR[2] and GILBERT LAPORTE[3,4]

[1] Télé-Université, Québec, [2] Université Laval, Québec, [3] Université de Montréal,
[4] École des Hautes Études Commerciales, Montréal, Canada

Solutions produced by the first generation of heuristics for the vehicle routeing problem are often far from optimal. Recent adaptations of local search improvement heuristics, like tabu search, produce much better solutions but require increased computing time. However there are situations where good solutions must be obtained quickly. The algorithm proposed in this paper yields solutions almost as good as those produced by tabu search adaptations, but at only a small fraction of their computing time. This heuristic can be seen as an improved version of the original petal heuristic. On 14 benchmark test problems, the proposed heuristic yields solutions whose values lie on average within 2.38% of that of the best known solutions.

Key words: petal method, sweep heuristic, vehicle routeing problem.

INTRODUCTION

The Vehicle Routeing Problem (VRP) holds a central place in distribution management. It can be defined as follows. Let $G = (V, A)$ be a graph where $V = \{v_0, \ldots, v_n\}$ is the vertex set and $A = \{(v_i, v_j): v_i, v_j \in V, i \neq j\}$ is the arc set. Vertex v_0 represents a *depot* at which are based m identical vehicles of capacity Q, while the remaining vertices correspond to customers or cities. With each vertex $v_i \in V$ is associated a non-negative demand q_i and a service time s_i. In the version of the problem under consideration, all arcs are undirected, i.e. they are *edges*. With each such edge (v_i, v_j) is associated a non-negative cost, c_{ij}, interpreted here as a travel time. The number of vehicles is not determined *a priori*. The VRP consists of determining a set of m vehicle routes (1) starting and ending at the depot, and such that (2) each customer is visited exactly once, (3) the total demand of any vehicle route does not exceed Q, (4) the duration of any route (including service times) does not exceed a preset upper limit D, and (5) the total cost of all routes is minimized.

The VRP is a hard combinatorial optimization problem which reduces to the Traveling Salesman Problem (TSP) when $m = 1$, and both Q and D are sufficiently large. Surveys are provided by Christofides[1], Laporte and Nobert[2], Laporte[3] and Fisher[4]. In the current state of knowledge, the VRP can rarely be solved to optimality for instances involving more than 50 customers. The best existing exact algorithms appear to be those of Cornuéjols and Harche[5], and of Christofides, Hadjiconstantinou and Mingozzi[6]. Fisher[7] also describes a branch and cut algorithm for the case where single-customer routes are forbidden. Several approximation algorithms have also been proposed. These include the classical savings algorithm[8], the sweep algorithm[9], the generalized assignment based heuristic of Fisher and Jaikumar[10], and some parallel insertion algorithms[11,12]. These methods are often one-phase heuristics that concentrate on building a feasible solution, with little effort spent on solution improvement. Recently, there have been significant developments in the area of local search improvement heuristics such as simulated annealing and tabu search. For a survey, see Gendreau, Laporte and Potvin[13]. Tabu search appears to be the most powerful method. Its application to the VRP by Osman[14], Taillard[15] and Gendreau, Hertz and Laporte[16] has produced the best known results on the fourteen benchmark problems described by Christofides, Mingozzi and Toth[12].

Correspondence: G. Laporte, Centre de Recherche sur les Transports, Université de Montréal, Case postale 6128, succursale centre-ville, Montréal, Quebec, Canada H3C 3J7

The recent results obtained with tabu search indicate that solutions produced by the first generation of heuristics are often far from optimal and deviations from optimality are typically in the 10–15% range. While the solutions produced by tabu search are much better (and sometimes optimal), these require substantial computing times and several parameter settings. For example, on the fourteen benchmark problems, Osman's[14] algorithm takes between 0.83 and 93 minutes on a VAX 8600 computer, while Gendreau, Hertz and Laporte[16] report computing times ranging from 6 to 100 minutes on a Silicon Graphics workstation. Taillard[15] uses parallel computing and does not report computational times.

When problems must be solved infrequently and significant sums of money are at stake, it makes sense to spend large amounts of time on their solution. There are situations, however, where good quality solutions, must be produced quickly, often in real time. This is the case, for example, of several families of dynamic and stochastic VRPs where route reoptimization is allowed to take place as information is gathered on the value of stochastic variables (e.g. random demands[17]). The algorithm we propose yields solutions nearly as good as those produced by the best available heuristics (e.g. tabu search), at only a fraction of the computing time.

We describe the algorithm in the next section, followed by the computational results and by the conclusion.

ALGORITHM

As with column generation algorithms (see, e.g. Balinski and Quandt[18], Agarwal, Mathur and Salkin[19], and Desrochers, Desrosiers and Solomon[20]), the proposed heuristic generates a set of good vehicle routes and then makes an optimal selection of some of these routes using a set partitioning algorithm. Other algorithms belonging to the same family are the *sweep* algorithm of Gillett and Miller[9], and the *petal* method originally proposed by Foster and Ryan[21] and extended by Ryan, Hjorring and Glover[22]. More specifically, we generate a number of r-petals, i.e. sets of r routes which collectively service all customers contained within a given sector centred at the depot (Figure 1).

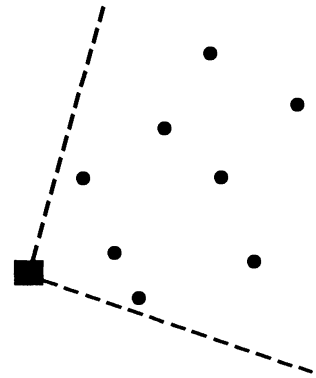

FIG. 1. *Customers contained within a given sector.*

The proposed heuristic generates 1-petals and 2-petals using a sweep procedure (Figures 2 and 3). It applies to each of them a routeing heuristic to check the feasibility of the proposed itinerary and to compute an upper bound on its optimal cost.

In what follows, we first describe how the itineraries are generated. We then show how these modules are integrated within a global route generation procedure. Finally, we outline the petal selection procedure.

1-Petal heuristic

In the case of 1-petals, or single vehicle routes, between the depot and a set S of customers, we use the TSP heuristic I^3 recently developed by the authors[23]. I^3 is a three-phase heuristic that

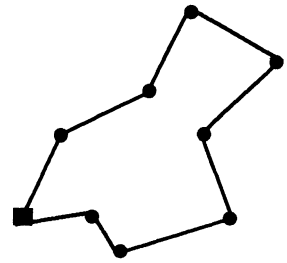

FIG. 2. *A 1-petal.*

constructs a good Hamiltonian cycle over a set of vertices by first forming an initial envelope (not necessarily convex) of the vertices, then sequentially inserting the remaining vertices into the partially constructed tour, and finally improving the resulting tour by means of a restricted 4-opt edge exchange mechanism called 4-opt*. On randomly generated problems, I^3 dominates some of the best composite heuristics for the TSP. On 100-vertex instances, it produces solution values on the average within 2.2% of the optimum. As the size of the single routes in our test problems will be considerably less, we should expect near-optimal solutions.

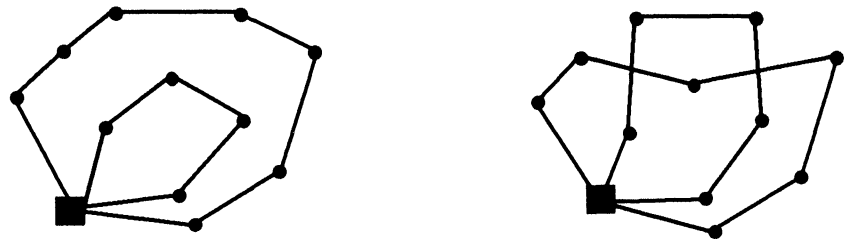

FIG. 3. *2-Petals*

2-Petal heuristic

To construct 2-petals over a set of vertices S, we first consider two seed vertices and create two back and forth routes between the seeds and the depot. Each remaining vertex is then included in the two routes using a cheapest insertion criterion, while maintaining feasibility. A reoptimization of the partial routes is performed using the 4-opt* exchange procedure[23] each time the number of vertices on a route is a multiple of an input parameter γ. An edge exchange procedure to be described later is also applied whenever some vertices cannot be inserted into one of the two routes.

We select as seeds the two vertices that are the furthest apart in S. Other simple rules were applied, e.g. considering the angle between the two vertices and the depot, but none produced better results than the largest distance rule. Several insertion rules were also attempted. The simplest one is to compute the insertion cost for each of the two routes of each unrouted vertex, and implement the cheapest feasible insertion. The disadvantage of this procedure is that it often creates unbalanced routes at the initial stages, resulting in highly suboptimal routes if the problem is tightly constrained. To counter this, we define a tightness coefficient $\alpha = 2Q - \sum_{v_i \in S} q_i$ to guide the search process. Whenever an insertion is attempted, we impose that the absolute difference between the total demand of each route should not exceed α. Thus, if $\sum_{v_i \in S} q_i$ is small, α will be large and will have little effect on the insertion process; however, if $\sum_{v_i \in S} q_i$ is larger, α will be small and will force the creation of balanced routes at each iteration. When this condition cannot be respected, we simply use the cheapest insertion. Since our problem may involve a maximal duration constraint, the current length of a route may at some point exceed D. To help counter this, each of the two routes is reoptimized when its number of vertices reaches a new multiple of the input parameter γ. We use for this the 4-opt* edge exchange procedure[23].

If some vertices of S remain uninserted after applying the above procedure, the following improvement mechanism may be applied to increase the likelihood of inserting more vertices. Let (i_h, j_h, k_h, l_h) be a sequence of four vertices (any of these vertices may be the depot) on each of the two routes $h = 1$ and $h = 2$. Then the following moves are attempted, but the depot is never moved: (1) insert j_1 between i_2 and j_2; (2) insert j_2 between i_1 and j_1; (3) swap j_1 and j_2; (4) insert (j_1, k_1) between i_2 and j_2, considering the two possible orientations; (5) insert (j_2, k_2) between j_1 and k_1, considering the two orientations; (6) swap (j_1, k_1) with (j_2, k_2), considering all four combinations. No move is executed unless it produces a better feasible solution. As soon as a profitable move is identified, it is implemented and the insertion process is restarted. Note that these operations constitute a restriction of the so-called λ-interchange mechanism described by Osman[14]. Extensive tests have shown that using this limited set of moves results in almost no loss of quality with respect to all possible moves. This can be explained in part by the fact that very often, the 4-opt* procedure will have been applied to these routes and little room is left for improvement.

This improvement procedure is also applied to the final 2-petal over S. However, to save time, it is only applied if the problem over the current 2-petal is tightly constrained, i.e. if $\sum_{v_i \in S} q_i > (1 + \beta)Q$, where $\beta \in [0,1]$ is a user controlled parameter, or if $d(S) > (1 + \beta)D$, where $d(S)$ is the current total duration of the two routes defined over vertex set S. Thus, if $\beta = 0$, the procedure is always applied and if $\beta = 1$, it is never applied.

When all vertices of S have been included in one of two feasible routes, reoptimize each of them using the 1-petal heuristic.

Generating all 1-petals and 2-petals

We are now in a position to describe the procedure used for the generation of all 1-petals and 2-petals. Without loss of generality, assume vertices are relabeled in increasing order of the polar angle they form with an arbitrary radius centred at the depot; ties are broken by first selecting the vertex having the smallest radius. In what follows, assume $v_j = v_{j(\text{mod } n)}$ if $j > n$. Petals are constructed by applying the following steps.

Step 1 (Initialization). Set $i := 0$.

Step 2 (Termination check). Set $i := i + 1$. If $i > n$, stop

Step 3 (Route initialization). Consider the customer set $S_{ii} := \{v_i\}$; record it and the cost $\bar{c}_{ii} := 2c_{0i}$ of the associated route. Set $j := i + 1$. Generate the customer set $S_{ij} := \{v_i, v_j\}$ and consider the route (v_0, v_i, v_j, v_0). If it is infeasible, go to Step 5. Otherwise, record S_{ij}, the associated route, and its cost $\bar{c}_{ij}$.

Step 4. (1-petal expansion). Set $j := j + 1$ and $S_{ij} := \{v_i, \ldots, v_j\}$. If the total demand of S_{ij} exceeds Q, go to Step 5. Otherwise, apply the 1-petal heuristic with $S := S_{ij}$. If no feasible route can be identified, go to Step 5. Otherwise, record S_{ij}, the associated solution, and its cost $\bar{c}_{ij}$. Repeat this step.

Step 5. (2-petal expansion). If the total demand of S_{ij} exceeds $2Q$, go to Step 6. Otherwise, apply the 2-petal heuristic with $S := S_{ij}$. If no feasible 2-petal can be identified, go to Step 6. Otherwise, record S_{ij}, the associated solution, and its cost $\bar{c}_{ij}$. Set $j := j + 1$ and $S_{ij} := \{v_i, \ldots, v_j\}$. Repeat this step.

Step 6. (Dominance test). Some of the petals just created may be dominated. If $j = 2$, go to Step 2. Otherwise, for $h = j, j - 1, \ldots, 3$, consider the vertex set S_{ih} and the last vertex v_h inserted in S_{ih}. Let $\bar{c}_{ih}$ be the cost of the corresponding petal. If $\bar{c}_{i,h-1} \geqslant \bar{c}_{ih}$, removing v_h from S_{ih} results in a petal whose cost does not exceed $\bar{c}_{i,h-1}$. The petal defined in $S_{i,h-1}$ is then dominated and replaced by the petal defined over $S_{ih} \backslash \{v_h\}$. Go to Step 2.

One advantage of this petal generation approach, as opposed to the original petal method[21], is that it enables the creation of embedded or intersecting vehicle routes often encountered in optimal solutions (Figure 4).

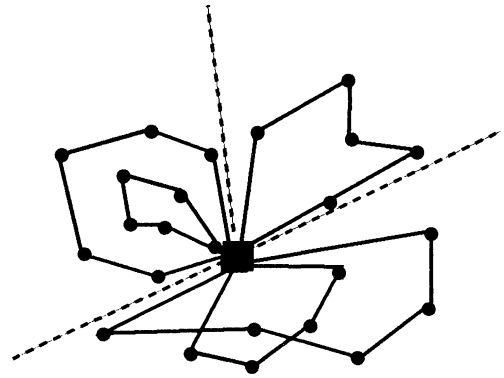

FIG. 4. *A possible solution produced by the heuristic.*

Petal selection procedure

Once all 1-petals and 2-petals have been generated, an optimal combination can be determined by solving a set partitioning problem of the form

$$\text{Minimize} \sum_{l \in L} \bar{c}_l x_l$$

subject to

$$\sum_{l \in L} a_{kl} x_l = 1 \quad (k = 1, \ldots, n)$$

$$x_l = 0 \text{ or } 1 \quad (l \in L),$$

where L is the set of candidate r-petals ($r = 1$ or 2), $\bar{c}_l$ is the cost of petal l, and $a_{kl} = 1$ if and only if vertex v_k belongs to petal l. Since each petal corresponds to a contiguous sector of vertices, the (a_{kl}) matrix possesses the column circular property (i.e. the 1's in each column are consecutive modulo n) and the set partitioning problem can be solved in polynomial time[22] by reduction to a series of shortest path problems in an acyclic graph (see, e.g. Beasley[24] and Ulusoy[25]). A detailed description of this set partitioning algorithm is provided in Boctor and Renaud[26].

Route merge

A final check is performed over all pairs of vehicle routes just selected to determine whether any of these can be feasibly combined. If so, the merge is implemented, and the combined route is reoptimized using the 1-petal heuristic.

COMPUTATIONAL RESULTS

The improved petal algorithm just described was tested on the fourteen benchmark problems described in Christofides, Mingozzi and Toth[12], using the parameter values $\gamma = 5$ and $\beta = 0.25$. Comparisons were made with the classical sweep algorithm of Gillett and Miller[9] and with a 1-petal generation algorithm of the type described by Foster and Ryan[21]. We also provide comparisons with the best known results produced by Taillard[15]. As this author does not provide computation times, we only report his solution values.

Before we proceed with the presentation of the results, a few words of caution are in order. First, straight comparisons with results reported in the literature are often misleading. Computation times cannot always be compared because of the different computers and operating systems involved. Also, the rounding and truncating conventions often vary from author to author: final solution values can differ significantly according to whether real, rounded or truncated c_{ij}'s are used. On this subject, see Mole[27] and Gendreau, Hertz and Laporte[16]. Finally, there are several ways to code the sweep or 1-petal algorithms as the rules are not always well specified in the

original articles. As a result, the outcome is related to the implementation. All our programs were coded in Pascal with the same data structures, and executed on a Sun Sparcstation 2 (28.5 Mips, 4.2 Mflops) with 32 megabytes of Ram. All tests were executed using real c_{ij}'s, and the final solution value was rounded up or down after two decimals.

For our implementation of the sweep algorithm, we followed the instructions provided in Christofides, Mingozzi and Toth[12] and made the following choices whenever these instructions were ambiguous: (1) when several vertices have the same polar angle, ties are again broken by first selecting the vertex with the smallest radius; (2) in problems with capacity constraints only, the 'emerging route R' is reoptimized whenever the vehicle capacity is attained. This is done using 3-opt, where the starting tour corresponds to the sequence in which the vertices were introduced in the sweep process. In problems with maximal duration constraints, vertices are gradually introduced using a cheapest insertion criterion, and 3-opt reoptimization takes place whenever the capacity or duration constraint is violated. In the latter case, it may be possible to include additional vertices in the route; (3) the set U was formed by taking the next five unrouted vertices; (4) the value of $D2$ is found using 3-opt; (5) when vertices are swapped between R and U, each route is reoptimized using 3-opt, starting with the current routes.

Similarly, the petal method is not fully specified in the original article[21]. We implemented it as in our improved petal algorithm, by simply skipping the generation of 2-petals.

Our computational results are summarized in Table 1. We first note that our results reported for the sweep algorithm are comparable, but slightly better on the average than those reported by Christofides, Mingozzi and Toth[12]. No basis for comparison is available for the original 1-petal algorithm. Our results indicate that in terms of solution quality, the improved petal algorithm is superior to the sweep heuristic of Gillett and Miller[9] and to our implementation of the 1-petal algorithm. On average, our solution values lie within 2.38% of the best known values. Note that the sweep algorithm is regarded as one of the best construction heuristics available[12,16]. The 1-petal algorithm is much faster than sweep, mostly because it does not use the time consuming 3-opt procedure. The improved petal heuristic never takes more than 12 minutes, but is more than ten times slower than the 1-petal algorithm due to the effort spent on the generation 2-petals. The payoff is increased accuracy: our solutions are 2.38% above the best known values, instead of 5.85%. Finally, note that the worst deviation recorded with improved petal is 6.43%, compared with 21.45% for sweep and 20.22% for 1-petal.

CONCLUSIONS

We have described a new heuristic for the VRP that incorporates features of the sweep algorithm first described by Gillett and Miller[9] and of column generation procedures[18–20]. It can also be viewed as an improved version of the original petal algorithm[21]. On fourteen classical test problems, our algorithm yields near-optimal solutions within very modest computing times. It constitutes a good compromise between simple algorithms such as sweep[9], and better but more time consuming tabu search based algorithms[14–16]. Further improvements could conceivably be achieved by generating, in addition to 1-petals and 2-petals, r-petals with $r \geqslant 3$, at the expense of significantly increased computing time. Note that practitioners often dislike routes that intersect too frequently and therefore, limiting ourselves to $r \leqslant 2$ is probably a good compromise. Finally, since the proposed approach is quite fast, it can realistically be embedded within a local search heuristic such as tabu search.

Acknowledgements—This work was partly supported by the Canadian Natural Sciences and Engineering Research Council under grants OGP 0036509 and OGP 0039682. This support is gratefully acknowledged. Thanks are also due to two anonymous referees for their valuable comments.

TABLE 1. *Summary of computational results*

Problem[1]	n	Type[2]	Sweep			Petal			Improved petal			Taillard
			Cost	% Above best	Time[3]	Cost	% Above best	Time[3]	Cost	% Above best	Time[3]	Cost[4]
1	50	C	531.90	1.01	0.12	531.90	7.01	0.10	524.61	0	0.76	524.61[5]
2	75	C	884.20	5.86	0.17	885.02	5.96	0.07	854.09	2.25	0.52	835.26
3	100	C	846.34	2.44	1.18	836.34	1.23	0.32	830.40	0.51	3.84	826.14
4	150	C	1075.38	4.57	2.53	1070.50	4.09	0.41	1054.62	2.55	5.93	1028.42
5	199	C	1396.05	7.49	3.60	1406.84	8.32	0.41	1354.23	4.27	6.21	1298.79
6	50	C, D	560.08	0.84	0.16	560.08	0.84	0.09	560.08	0.84	0.56	555.43
7	75	C, D	965.51	6.14	0.19	968.89	6.51	0.07	922.75	1.44	0.43	909.68
8	100	C, D	883.56	2.03	1.47	877.80	1.37	0.25	877.29	1.31	2.91	865.94
9	150	C, D	1220.71	5.00	3.00	1220.20	4.96	0.26	1194.51	2.75	3.58	1162.55
10	199	C, D	1526.64	9.21	4.91	1515.95	8.44	0.35	1470.31	5.18	5.19	1397.94
11	120	C	1265.65	21.45	3.52	1252.84	20.22	0.61	1109.14	6.43	11.70	1042.11
12	100	C	919.51	12.20	0.64	824.77	0.64	0.21	824.77	0.64	2.11	819.56
13	120	C, D	1785.30	15.84	2.24	1773.69	15.09	0.26	1585.20	2.86	3.31	1541.14
14	100	C, D	911.81	5.24	0.85	894.77	3.28	0.17	885.87	2.25	1.69	866.37
Average				7.09	1.76		5.85	0.26		2.38	3.48	

[1] In problems 1 to 10, vertices are randomly generated in the plane. In problems 11 to 14, vertices are clustered.
[2] C: capacity restriction; D: distance restriction.
[3] Minutes on a Sun Sparc 2 workstation.
[4] Best known solution value: see Taillard[15]
[5] Optimal solution value: see Christofides, Hadjiconstantinou and Mingozzi[6].

REFERENCES

1. 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.
2. G. Laporte and Y. Nobert (1987) Exact algorithms for the vehicle routing problem. *Ann. Discrete Maths.* **31**, 147–184.
3. G. Laporte (1992) The vehicle routing problem: An overview of exact and approximate algorithms. *Eur. J. Opal Res.*, **59**, 345–358.
4. M. L. Fisher (1995) Vehicle routing. In *Networks Routing* (M. O. Ball, T. L. Magnanti, C. L. Monma and G. L. Nemhauser, Eds) pp 1–33. North-Holland, Amsterdam.
5. G. Cornuéjols and F. Harche (1993) Polyhedral study of the capacitated vehicle routing problem. *Math. Prog.* **60**, 21–52.
6. N. Christofides, E. Hadjiconstantinou and A. Mingozzi (1995) A new exact algorithm for the vehicle routing problem based on q-path and k-shortest path relaxations. *Ann. Opns Res.* Forthcoming.
7. M. L. Fisher (1994) Optimal solution of vehicle routing problems using minimum K-trees. *Opns Res.* **42**, 626–642.
8. G. Clarke and J. W. Wright (1964) Scheduling of vehicles from a central depot to a number of delivery points. *Opns Res.* **46**, 93–100.
9. B. Gillett and L. Miller (1974) A heuristic for the vehicle dispatching problem. *Opns Res* **22**, 340–349.
10. M. L. Fisher and R. Jaikumar (1981) A generalized assignment heuristic for vehicle routing. *Networks*, **11**, 109–124.
11. K. Altinkemer and B Gavish (1991) Parallel savings based heuristics for the delivery problem. *Opns Res.* **39**, 456–469.
12. N. Christofides, A. Mingozzi and P. Toth (1979) The vehicle routing problem. In *Combinatorial Optimization.* (N. Christofides, A. Mingozzi and P. Toth, Eds.) pp. 315–338. Wiley, Chichester.
13. M. Gendreau, G. Laporte and J.-Y. Potvin (1994) Local search algorithms for the vehicle routing problem. Working paper CRT-963, Center for research on transportation, Université de Montréal, Canada.
14. I. H. Osman (1993) Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Ann. Opns Res.* **41**, 421–451.
15. E. Taillard (1993) Parallel iteration search methods for vehicle routing. *Networks*, **23**, 661–673.
16. M. Gendreau, A. Hertz and G. Laporte (1994) A tabu search heuristic for the vehicle routing problem. *Mgmt Sci* **40**, 1276–1290.
17. M. Gendreau, G. Laporte and R. Séguin (1995) The vehicle routing problem with stochastic customers and demands. *Transportation Science* **29**, 143–155.
18. M. Balinski and R. Quandt (1964) On an integer program for a delivery problem. *Opns Res.* **12**, 300–304.
19. Y. Agarwal, K. Mathur and H. M. Salkin (1989) A set-partitioning-based algorithm for the vehicle routing problem. *Networks*, **19**, 731–750.
20. M. Desrochers, J. Desrosiers and M. M. Solomon (1992) A new optimization algorithm for the vehicle routing problem with time windows. *Opns Res.* **40**, 342–354.
21. B. A. Foster and D. M. Ryan (1976) An integer programming approach to the vehicle scheduling problem. *Opl Res. Q.* **27**, 307–384.
22. D. M. Ryan, C. Hjorring and F. Glover (1993) Extension of the petal method for vehicle routing. *J. Opl Res. Soc.* **44**, 289–296.
23. J. Renaud, F. F. Boctor and G. Laporte (1996) A fast composite heuristic for the symmetric traveling salesman problem. *INFORMS J. on Computing* (forthcoming).
24. J. E. Beasley (1983) Route first-cluster second methods for vehicle routing. *Omega*, **11**, 403–408.
25. G. Ulusoy (1985) The fleet size and mix problem for capacitated arc routing. *Eur. J. Opl Res.* **22**, 329–337.
26. F. F. Boctor and J. Renaud (1993) Optimal solution of column-circular set-partitioning problems. Working Paper 93-27, Faculté des sciences de l'administration, Université Laval, Canada.
27. R. H. Mole (1983) The curse of unintended rounding error: a case from the vehicle scheduling literature. *J. Opl Res. Soc.* **34**, 607–613.

Received June 1994; accepted June 1995 after one revision