Journal of the Operational Research Society (2002) 53, 512–522 ©2002 Operational Research Society Ltd. All rights reserved. 0160-5682/02 $15.00

www.palgrave-journals.com/jors

A guide to vehicle routing heuristics

J-F Cordeau,[1] M Gendreau,[2] G Laporte,[1]* J-Y Potvin[2] and F Semet[3]

[1]*École des Hautes Études Commerciales, Montréal, Canada;* [2]*Université de Montréal, Montréal, Canada; and* [3]*Université de Valenciennes et du Hainaut Cambresis, Valenciennes, France*

Several of the most important classical and modern heuristics for the vehicle routing problem are summarized and compared using four criteria: accuracy, speed, simplicity and flexibility. Computational results are reported.

Journal of the Operational Research Society (2002) **53**, 512–522. DOI: 10.1057/palgrave/jors/2601319

Keywords: vehicle routing problem; heuristics

Introduction

The *Vehicle Routing Problem* (VRP), introduced by Dantzig and Ramser[1] in 1959, holds a central place in distribution management and has become one of the most widely studied problems in combinatorial optimization. The *Classical VRP* can be formally defined as follows. Let $G = (V, A)$ be a graph where $V = \{v_0, v_1, \ldots, v_n\}$ is a vertex set, and $A = \{(v_i, v_j) : v_i, v_j \in V, i \neq j\}$ is an arc set. Vertex v_0 represents a depot, while the remaining vertices correspond to customers. With A are associated a cost matrix (c_{ij}) and a travel time matrix (t_{ij}). If these matrices are symmetrical, as is commonly the case, then it is standard to define the VRP on an undirected graph $G = (V, E)$, where $E = \{(v_i, v_j) : v_i, v_j \in V, i < j\}$ is an edge set. Each customer has a non-negative demand q_i and a service time t_i. A fleet of m identical vehicles of capacity Q is based at the depot. The number of vehicles is either known in advance or treated as a decision variable. The VRP consists of designing a set of at most m delivery or collection routes such that (1) each route starts and ends at the depot, (2) each customer is visited exactly once by exactly one vehicle, (3) the total demand of each route does not exceed Q, (4) the total duration of each route (including travel and service times) does not exceed a preset limit D, and (5) the total routing cost is minimized. A common variant is where a time window $[a_i, b_i]$ is imposed on the visit of each customer. Several other extensions have also been studied; the vehicle fleet may be heterogeneous,[2] vehicles may perform both collections and deliveries on the same route,[3] some vehicles may be unable to visit certain sites,[4] some customers may require several visits over a given time period,[5] there may exist more than one depot,[5] deliveries may be split among several vehicles,[6] etc. For overview articles on the VRP, see Golden and Assad,[7] Fisher,[8] Desrosiers *et al*,[9] Crainic and Laporte,[10] Toth and Vigo,[11] Laporte and Semet,[12] Gendreau *et al*[13] and Cordeau *et al.*[14]

The VRP is a hard combinatorial optimization problem and only relatively small instances can be solved to optimality. To this day, it seems that no exact algorithm is capable of consistently solving instances in excess of 50 customers.[15] This is due to the fact that sharp lower bounds on the objective value are hard to derive, which means that partial enumeration based exact algorithms (using branch-and-bound or dynamic programming) will have a slow convergence rate. Since exact approaches are in general inadequate, heuristics are commonly used in practice.

There has been a steady evolution, over the past 40 years, in the development of VRP heuristics. In the early *classical heuristics*, much of the emphasis was put on quickly obtaining a feasible solution and possibly applying to it a postoptimization procedure. This class of methods includes the well-known savings algorithm,[16] the sweep algorithm[17] and the Fisher and Jaikumar algorithm.[18] Over the last ten years, much of the research effort has concentrated on the development of algorithms based on *metaheuristics*, using mainly two principles: *local search* and *population search*. In local search methods, an intensive exploration of the solution space is performed by moving at each step from the current solution to another promising solution in its neighbourhood. *Simulated annealing* (SA)[19] and *tabu search* (TS)[20] are two prime examples of this principle. Population search consists of maintaining a pool of good parent solutions and recombining them to produce offspring. A classical example is *genetic search* (GS)[21] which combines two parents to produce offspring. *Adaptive memory procedures* (AMPs)[22] can be viewed as an extension of GS where several parents are used to produce several offspring. While more time consuming than the early heuristics,

Correspondence: G Laporte, Centre de recherche sur les transports (C.R.T.), Campus de l'Université de Montréal, C.P. 6128, Succursale Centre-ville, Montréal H3C 3J7, Canada.

metaheuristics are capable of consistently producing high quality solutions. Thus, on the Christofides, Mingozzi and Toth (CMT)[23] 14 classical VRP benchmark instances, the most powerful methods have produced one proven optimum while the solutions obtained on the remaining instances are believed to be optimal or near-optimal since they have not been improved after thousands of hours of computing effort, using sophisticated search techniques.

Yet most commercial software and several in-house computer programs used by companies are based on unsophisticated methodologies, sometimes dating back to the 1960s. There are several reasons for this state of affairs. One is that the optimization component of VRP software is only a small part of the product, most of the effort being expanded on data management and sophisticated user interfaces. It is not uncommon for routing experts to use low quality solutions produced by VRP software and exploit the interactive capabilities of the system to perform manual improvements. Another reason is that company analysts and software developers are simply unaware of the latest algorithmic developments. While there is some truth to this, we believe the main hindrance to technology transfer may be more deep-rooted. In this paper, we argue that most of the available VRP heuristics lack some of the necessary attributes to ensure their adoption by practitioners.

In the following section we describe what we believe are four essential attributes for software transferability and end-user adoption. We then provide an appraisal of some of the best known heuristics with respect to these criteria. As much as we have tried to base this study on published and objective data, our analysis is at times personal and critical, especially with respect to the more intangible criteria. The algorithms are grouped in two categories: classical heuristics and metaheuristics. A summary and conclusion close the paper.

Four attributes of good VRP heuristics

Vehicle routing heuristics, as are most heuristics, are usually measured against two criteria: *accuracy* and *speed*. In our opinion *simplicity* and *flexibility* are also essential attributes of good heuristics. We now elaborate on these four criteria.

Accuracy

Accuracy measures the degree of departure of a heuristic solution value from the optimal value. Since optima and sharp lower bounds are usually unavailable in the case of the VRP, most comparisons have to be made with best known values. As pointed out by Barr *et al*,[24] analysing heuristic results is fraught with difficulties. Authors often report results obtained for the best combination of algorithmic parameters, or for the best of several runs with different starting solutions. Not all authors use the same rounding or truncating conventions which, as emphasized by Gendreau *et al*,[25] can lead to vastly different results. Tests are often carried out by rounding costs d digits after the decimal point, ie, by setting $c_{ij} := 10^{-d}\lfloor 10^d c_{ij} + 0.5 \rfloor$, where $\lfloor y \rfloor$ is the integer part of y. More rarely, costs are truncated d digits after the decimal point, ie $c_{ij} := 10^{-d}\lfloor 10^d c_{ij} \rfloor$. Tests can also be performed with floating point arithmetic, without rounding or truncating, using as many digits as allowed by the computer. When rounding or truncating occurs, it is common to recompute the final solution cost using more than d digits after the decimal point and to present the final result with one or two digits after the point. To illustrate the discrepancy in results, consider the first instance of the CMT test problems. The same tabu search algorithm (Taburoute) yields a cost of 524.61 if computations are performed with rounded costs c_{ij} and $d = 5$ (this is, by the way, a proven optimum[26]), a cost of 521 if costs are rounded up or down to the nearest integer, and a cost of 508 if truncated integer costs are used. In instances with route duration constraints, using an insufficient precision may yield an infeasible solution if travel times are proportional to distances. For a further discussion on the bias introduced by rounding in VRPs, see Mole.[27]

Another issue related to accuracy is consistency. As a rule, users will prefer a heuristic that performs well all the time rather than one that may perform even better most of the time but very poorly on other occasions, and may even produce solutions easily perfectible by visual inspection. Such solutions are enough to discredit the algorithm.

Finally, users will often prefer an algorithm that produces a good solution at an early stage, and then displays solutions of increasing quality throughout the execution to an algorithm that comes up with only a final answer, possibly after a long computing time. This gives users a better feel of how much additional effort is worth investing given the evolution rate of the solution value.

Speed

Just how important is computation speed in vehicle routing? It all depends on the planning level at which the problem is solved and on the degree of accuracy required. At one extreme, real-time applications such as express courier pickup and delivery[28] or ambulance redeployment[29] require fast, sometimes almost instantaneous, action. For example, Gendreau *et al*[29] describe the crucial role played by parallel computing in a setting where an ambulance relocation strategy must be determined every three minutes on average. At the other extreme, in long term planning decisions made every several months, such as fleet sizing, it makes sense to invest several hours or even several days of computing time, particularly if large sums of money are at stake. Most applications fall somewhere between these two extremes. It does not seem unreasonable to invest ten or twenty minutes of computing time on a routing problem that must be solved daily. Interactive systems must of course react much more quickly. The issue of speed is not always properly put into perspective. For example, would not most practitioners prefer a VRP heuristic that is 2.38% accurate and runs in 3.48 min to one that is 5.85% accurate but requires only 0.26 min of computing time? (These are actual statistics of the 2-petal and 1-petal algorithms described by Renaud *et al.*[30])

Accurate reporting of computing time is another delicate issue in the scientific literature, particularly in the case of multiple runs or if parallel computing is used. As in the area of accuracy, strict standards are not consistenly enforced by VRP researchers when it comes to assessing the speed of algorithms.

Simplicity

Several VRP heuristics are rarely implemented because they are just too complicated to understand and to code. While it is unrealistic to expect scientific articles to provide a minute description of every algorithmic detail, sufficient information should be provided to enable a reasonably skilled programmer to come up with a working code. In addition, heuristics should be reasonably robust to ensure that they work properly, even if not every single detail is implemented. Many algorithmic descriptions fail on the count of providing too much or insufficient detail. One reason why the Clarke and Wright[16] algorithm is so popular among practitioners is that its basic principle is trivial to understand and easy to code. (Amazingly, the original description is rather clumsy and would probably fall short of today's publishing standards.) Simple codes, preferably short and self-contained, stand a better chance of being adopted, although a minimum of complexity is to be expected for good results.

Algorithms that contain too many parameters are difficult to understand and unlikely to be used. This problem is prevalent in most metaheuristics developed over the past ten years. In their quest for ever better solutions, researchers have increased the number of parameters contained in their algorithms far beyond what can be deemed reasonable, particularly in view of the fact that relatively few instances are used in the tests. This problem was recently raised by Golden *et al.*[31] Not only should the number of algorithmic parameters be limited, but these should also make sense to the end-user. Thus a parameter controlling the number of consecutive iterations without improvement in a local search heuristic is easy to understand, whereas a bound on the number of executions of an internal procedure is meaningless to most people.

There are two easy ways around the proliferation of parameters. One is to set them once and for all at some meaningful value, especially if tests show that the algorithm is rather insensitive to a particular parameter choice. Another possibility is to make use of parameters that self-adjust during the course of the algorithm.[5,25]

Flexibility

A good VRP heuristic should be flexible enough to accomodate the various side constraints encountered in a majority of real-life applications. While most of the VRP literature focuses on capacity and sometimes route length constraints, it is often clear how changes can be made to deal with additional constraints, but this is not always possible, and performance can also deteriorate significantly as a result. Our experience[5,25] suggests that an efficient way of handling side constraints in a local search process is through the use of two objectives. The first, $F(x)$, computes the routing cost of solution x. The second, $F'(x)$, is the sum of $F(x)$ and weighted penalty terms associated with violations of each side constraint. For example, if $Q(x)$ and $D(x)$ are the capacity and route duration violations associated with solution x, then $F'(x)$ would be defined as $F(x) + \alpha Q(x) + \beta D(x)$, where α and β are positive self-adjusting penalty parameters. Initially set equal to 1, these parameters are periodically increased or decreased throughout the search according to whether previous solutions were infeasible or feasible. This way of proceeding means that the search will probably evolve through a mix of feasible and infeasible solutions, thus reducing the probability of becoming trapped in a local minimum. Another algorithmic advantage of this device is that the search can operate with relatively simple moves, such as removing a customer from its current route and inserting it in a different route. When feasibility must be maintained at all cost, such simple moves tend not to work if side constraints are tight, and more complex and time consuming operations must then be envisaged, such as ejection chains.[32,33] So in a sense, algorithmic flexibility is in part achieved through simplicity of design.

Classical heuristics

Several heuristics have been devised for the VRP,[12] only some of which are sufficently well known to be truly viewed as 'classical'. For the sake of parsimony we concentrate on three of the best known heuristics: the Clarke and Wright algorithm, the sweep algorithm and the Fisher and Jaikumar algorithm. These were selected partly because of their popularity, and also because they operate on vastly different principles. Some extensions of these methods are also discussed in passing.

The Clarke and Wright savings heuristic

The Clarke and Wright (CW)[16] heuristic is one of the best known and remains widely used in practice to this day, despite some of its shortcomings. It is based on the notion of *saving*. Initially, a feasible solution consists of n back and forth routes between the depot and a customer. At any given iteration, two routes $(v_0, \ldots, v_i, v_0)$ and $(v_0, v_j, \ldots, v_0)$ are merged into a single route $(v_0, \ldots, v_i, v_j \ldots, v_0)$ whenever this is feasible, thus generating a saving $s_{ij} = c_{i0} + c_{0j} - c_{ij}$. In the *parallel* version of the algorithm, the merge yielding the largest saving is always implemented, whereas the *sequential* version keeps expanding the same route until this is no longer feasible. In practice the parallel version is much better. It is common to apply a 3-opt[34] post-optimization step to the final solution.

This algorithm scores very high on simplicity and speed. It contains no parameters and is easy to code. In our tests on the CMT benchmark instances, it typically ran within 0.12 s on a Sun Ultrasparc 10 workstation (440 MHz), without the 3-opt step. Executing a 3-opt post-optimization step increased the average running time to only 0.13 s. This algorithm obtains a medium score on accuracy. The best implementation (parallel version followed by 3-opt) produced an average deviation of 6.71% from the best known solution values identified by Taillard[35] and Rochat and Taillard[22] (see Table 1). In addition, several researchers have observed that the solution is sometimes of rather poor quality and often contains at least one rather circumferential route. The lack of flexibility is probably the worst feature of this algorithm. While additional constraints can, in principle, be incorporated in the CW algorithm, this usually results in a sharp deterioration in solution quality. This can be explained by the fact that the algorithm is based on a greedy principle and contains no mechanism to undo early unsatisfactory route merges. Solomon[36] reports a variant of the CW algorithm in which savings are adapted to handle time windows, but results are disappointing.

Several improvements to the CW algorithm have been proposed. Gaskell[37] and Yellow[38] have suggested using generalized savings of the form $s_{ij} = c_{i0} + c_{0j} - \lambda c_{ij}$ to help produce more compact routes, where λ is a positive parameter. Other enhancements are related to the use of sophisticated data structures and sorting strategies[39,40] to better handle the savings. We believe, however, that, given the present level of computer technology and the very fast running time of the CW algorithm on medium size instances, the latter improvements are fast becoming irrelevant.

Another stream of research on the CW algorithm has concentrated on optimising the route merging process through the use of a matching algorithm. Early results based on this idea were produced by Desrochers and Verhoog[41] and Altinkemer and Gavish.[42] The best and most recent implementation is due to Wark and Holt[43]

Table 1 Comparison of five classical heuristics for the VRP

			Clarke and Wright[a]		*Two-matching*[b]		*Sweep*[c]		*1-Petal*[c]		*2-Petal*[c]		
Instance	*n*	*Type*[d]	*Value*	*Seconds*[e]	*Value*	*Seconds*[f]	*Value*	*Seconds*[g]	*Value*	*Seconds*[g]	*Value*	*Seconds*[g]	*Best*
1	50	C	578.56	*0.03*	**524.6**[k]	*1200*	531.90	*7.2*	531.90	*6.0*	**524.61**	*45.6*	**524.61**[h,j]
2	75	C	888.04	*0.05*	835.8	*3000*	884.20	*10.2*	885.02	*4.2*	854.09	*31.2*	**835.26**[h]
3	100	C	878.70	*0.10*	830.7	*8700*	846.34	*70.8*	836.34	*19.2*	830.04	*230.4*	**826.14**[h]
4	150	C	1128.24	*0.21*	1038.5	*17100*	1075.38	*151.8*	1070.50	*24.6*	1054.62	*355.8*	**1028.42**[h]
5	199	C	1386.84	*0.32*	1321.3	*28800*	1396.05	*216.0*	1406.84	*24.6*	1354.23	*372.6*	**1291.45**[i]
6	50	C,D	616.66	*0.05*	**555.4**	*1800*	560.08	*9.6*	560.08	*5.4*	560.08	*33.6*	**555.43**[h]
7	75	C,D	974.79	*0.06*	911.8	*2700*	965.51	*11.4*	968.89	*4.2*	922.75	*25.8*	**909.68**[h]
8	100	C,D	968.73	*0.08*	878.0	*9900*	883.56	*88.2*	877.80	*15.0*	877.29	*174.6*	**865.94**[h]
9	150	C,D	1284.63	*0.17*	1176.5	*20700*	1220.71	*180.0*	1220.20	*15.6*	1194.51	*214.8*	**1162.55**[h]
10	199	C,D	1521.94	*0.33*	1418.3	*32100*	1526.64	*294.6*	1515.95	*21.0*	1470.31	*311.4*	**1395.85**[i]
11	120	C	1048.53	*0.14*	1043.4	*16500*	1265.65	*211.2*	1252.84	*36.6*	1109.14	*702.0*	**1042.11**[h]
12	100	C	824.42	*0.08*	**819.6**	*5700*	919.51	*38.4*	824.77	*12.6*	824.77	*126.6*	**819.56**[h]
13	120	C,D	1587.93	*0.12*	1548.3	*30600*	1785.30	*134.4*	1173.69	*15.6*	1585.20	*198.6*	**1541.14**[h]
14	100	C,D	868.50	*0.08*	**866.4**	*8400*	911.81	*5.1*	894.77	*10.2*	885.87	*101.4*	**866.37**[h]
Average deviation from best and time			6.71%	*0.13*	0.63%	*13371.42*	7.09%	*105.6*	5.85%	*15.6*	2.38%	*208.8*	

[a]Implemented by Laporte and Semet.[12] Verifiable results (detailed solutions are available in articles or on web sites). Computations performed by rounding costs seven digits after the decimal point.
[b]Implemented by Wark and Holt.[43] Verifiable results, best of five runs. Computations performed by rounding costs two digits after the decimal point.[43]
[c]Implemented by Renaud *et al.*[30] Verifiable results. Computations performed by rounding costs twelve digits after the decimal point.[66]
[d]C: capacity restriction, D: route duration restriction.
[e]Sun Ultrasparc 10 workstations (42 Mflops).
[f]Sun 4/630 MP. Total time for five runs.
[g]Sun Sparcstation 2 (210.5 Mips, 4.2 Mflops).
[h]Taillard.[35] Verifiable results.
[i]Rochat and Taillard.[22]
[j]Optimal solution value.
[k]Bold numbers correspond to best known values.

who often obtained significant improvements over the original CW implementation, but at the expense of much increased computing times. Their results, presented in Table 1, correspond to the best of five runs, each requiring on the average between 4 and 107 min on a Sun4/630MP. On the whole, this algorithmic enhancement removes from the CW heuristic two of its best features (speed and simplicity), is difficult to implement and does nothing to redress the lack of flexibility of the original algorithm. While the repeated application of a matching based algorithm yields increased accuracy (it produces average deviation of 0.63% on the CMT benchmark instances), this technique has in our opinion limited potential because of its complexity and low flexibility level.

The sweep algorithm

The sweep algorithm is generally attributed to Gillett and Miller[17] although its principle can be traced back to Wren[44] and Wren and Holliday.[45] It applies to planar instances of the VRP. Feasible routes are created by rotating a ray centred at the depot and gradually including customers in a vehicle route until the capacity or route length constraint is attained. A new route is then initiated and the process is repeated until the entire plane has been swept. A 3-opt step is then typically applied. On the CMT instances, the Renaud *et al*[30] implementation of this algorithm has yielded an average deviation of 7.09% from the best known solution values. Average computing times of 105.6 s were obtained on a Sun Sparcstation 2 (210.5 Mips, 4.2 Mflops) (see Table 1). This algorithm scores high on simplicity, but does not seem to be superior to CW both in terms of accuracy and speed. It is also rather inflexible. Again, the greedy nature of the sweep mechanism makes it difficult to accomodate extra constraints and the fact that the algorithm assumes a planar structure severely limits its applicability. In particular, the algorithm is not well suited to instances defined in an urban setting with a grid street layout.

A number of heuristics generate feasible vehicle routes (sometimes called *petals* in this context) and determine a best combination through the solution of a set partitioning problem. Prime examples of this approach are the 1-petal algorithm of Foster and Ryan[46] and Ryan *et al*[47] and the 2-petal heuristic of Renaud *et al*[30] where, in addition to single routes, double vehicle routes are also generated. These extensions provide accuracy gains with respect to the sweep algorithm. In terms of speed, improvements can also be made since the 3-opt step is no longer applied (see Table 1). These extensions do not get full marks on simplicity since generating a large pool of petals (especially 2-petals) can be cumbersome, and a set partitioning step must also be executed. As far as flexibility is concerned, petal algorithms can be made to accomodate a wide variety of constraints but this can come at the expense of simplicity. In fact, this type of algorithm can be viewed as a truncated version of column generation which is known to produce high quality results on tightly constrained VRPs with time windows,[48] but again simplicity is sacrificed.

The Fisher and Jaikumar algorithm

The Fisher and Jaikumar[18] algorithm is a two-phase process in which feasible clusters of customers are first created by solving a generalized assignment problem (GAP), and a vehicle route is determined on each cluster by means of a travelling salesman problem (TSP) algorithm. To formulate the GAP, it is necessary to first determine a *seed* for each route from which customer distances are computed. Since the GAP is NP-hard, it is usually solved by means of a Lagrangian relaxation technique. While good results were reported for this algorithm in the early 1980s we have some misgivings about its performance. The original article[18] provides integer solutions values without providing the rounding or truncating rule, and the solutions cannot be verified, which makes the assessment of the algorithm difficult. Our own computational experiments suggest that the Fisher and Jaikumar algorithm is not simple to program and its speed is highly related to the choice of seeds and to the implementation of Lagrangian process. We found that if seeds are selected as in Fisher *et al*[49] and the Lagrangian steps programmed as recommended by Fisher *et al*[50] then convergence is often poor, several trials may be necessary to reach a satisfactory solution, and even then accuracy can be low. We did not, in general, succeed in obtaining results close to those of Fisher and Jaikumar.[18] Flexibility is also problematic. While it is, in principle, simple to handle capacity constraints, the original reference contains no indication regarding the treatment of route duration constraints, although six instances involving such constraints are reportedly solved. We see no easy mechanism by which to incorporate these or other side constraints within the GAP algorithm. Bramel and Simchi-Levi[51] have optimized the choice of seeds in the Fisher and Jaikumar algorithm by solving a capacitated location problem. Their results on the seven CMT instances containing only capacity constraints show a significant average deviation (3.29%) from the best known results (see Table 2).

Metaheuristics

Compared with classical heuristics, metaheuristics perform a much more thorough search of the solution space, allowing inferior and sometimes infeasible moves, as well as recombinations of solutions to create new ones. This area of research has experienced a formidable growth over the past ten years and has produced some highly effective and flexible VRP heuristics.[13] It is fair to say, however, that the gains in solution quality obtained with these modern heuristics have often been made at the expense of speed and simplicity, although this is not the case of some of the more

Table 2 The Fisher and Jaikumar algorithm and the Bramel and Simchi-Levi enhancement

			Fisher and Jaikumar[a]		*Location-based heuristic*[b]		
Instance	*n*	*Type*[c]	*Value*	*Seconds*[d]	*Value*	*Seconds*[e]	*Best*
1	50	C	524	*9.3*	**524.6**[f]	*68*	**524.61**[g,i]
2	75	C	857	*12.0*	848.2	*406*	**835.26**[g]
3	100	C	833	*17.7*	832.2	*400*	**826.14**[g]
4	150	C	1014	*33.6*	1088.6	*2552*	**1028.42**[g]
5	199	C	1420	*40.1*	1461.2	*4142*	**1291.45**[h]
6	50	C,D	560	*15.2*	—	—	**555.43**[g]
7	75	C,D	916	*20.6*	—	—	**909.68**[g]
8	100	C,D	885	*52.2*	—	—	**865.94**[g]
9	150	C,D	1230	*121.3*	—	—	**1162.55**[g]
10	199	C,D	1518	*136.6*	—	—	**1395.85**[h]
11	100	C	—	—	1051.5	*1303*	**1042.11**[g]
12	120	C	824	*6.4*	826.1	*400*	**819.56**[g]
13	100	C,D	—	—	—	—	**1541.14**[g]
14	120	C,D	848	*6.3*	—	—	**866.37**[g]
Average deviation from best and time				*47.13*	3.29%	*1324.43*	

[a]Implemented by Fisher and Jaikumar.[18] Unverifiable results. The rounding or truncating rule is not reported.
[b]Implemented by Bramel and Simchi-Levi.[51] Verifiable results. Computations performed with double-precision floating point arithmetic.[67]
[c]C: capacity restrictions, D: route duration restrictions.
[d]DEC-10 computer.
[e]RS6000, Model 550.
[f]Bold numbers correspond to best known values.
[g]Taillard.[35]
[h]Rochat and Taillard.[22]
[i]Optimal solution value.[26]

recent implementations. The past decade has been very rich in algorithmic design and testing. Some promising ideas did not translate into equally good algorithms while others have withstood the test of time.

Tabu search (TS) clearly stands out as the best metaheuristic for the VRP. By and large, the best TS implementations for the VRP dominate other search processes such as simulated[52] and deterministic[53,54] annealing, genetic search,[55] ant systems[56] and neural networks[57] (see, eg, Gendreau *et al*[13]). The idea behind TS is to perform a local search by moving, at iteration t, from a solution x_t to the best solution x_{t+1} in its neighbourhood. Since moving from x_t to x_{t+1} may cause the objective function to deteriorate, an anti-cycling mechanism is put in place, namely any solution possessing some attribute of x_t is declared *tabu*, or forbidden, for a number of iterations. So, in effect, the best neighbour x_{t+1} of x_t is only selected if it is non-tabu or if it is clear cycling will not occur. Several additional mechanisms, such as diversification and intensification, have been implemented by a variety of researchers. Of course, not all TS heuristics for the VRP have been equally successful. The first known implementation, by Willard,[58] did not make use of a sufficiently powerful neighbourhood structure to allow the identification of high quality solutions. In contrast, several implementations that followed contained too many devices and user-controlled parameters. There has been a tendency in recent years toward leaner implementations. In what follows, we present some of the best available TS heuristics for the VRP.

Taburoute

With respect to earlier TS implementations, the Taburoute heuristic of Gendreau *et al*[25] is rather involved and contains several innovative features. The neighbourhood of x_t is the set of all solutions reachable from x_t by removing a vertex v from its current route r and inserting it in another route s containing one of its closest neighbours by means of a generalized insertion (GENI) procedure.[59] Reinserting v in route r is then declared tabu for θ iterations, where θ is randomly drawn from the interval [5, 10], as suggested by Taillard[60] in the context of the quadratic assignment problem. Intermediate infeasible solutions are considered and penalized through the use of self-adjusting penalty parameters, as explained earlier. Periodic reoptimisations of the individual vehicle routes are performed by means of the GENIUS heuristic[59] for the TSP. A continuous diversification strategy is also applied to penalise frequently moved vertices. This is done by adding to the objective function a term proportional to the relative past movement frequency of the vertex currently being considered. False starts are also employed: this allows a limited search starting from several initial solutions, and a full search is then performed using the most promising starting point. As is commonly done, the procedure ends when the objective has not improved for a number of consecutive iterations.

On the CMT test problems, Taburoute has produced high quality solutions: the average deviation from the best known values is 0.86% and five best known solutions were produced. Running times can be described as good but not excellent. Solution times vary between 6 and 100 min on Silicon Graphics workstation (36 MHz, 5.7 Mflops). Taburoute probably scores rather low on simplicity, partly due to its large number of user-controlled parameters (nine in all) and to its use of the outside TSP heuristic GENIUS which is not that easy to reproduce. Flexibility, however, is relatively high given that additional constraints can easily be incorporated through the penalty mechanism.

The Taillard tabu search algorithm

The Taillard[35] TS implementation was developed at the same time as Taburoute. It also uses random tabu durations and continuous diversification. Where it differs most from Taburoute is in the neighbourhood structure. Standard vertex insertions and exchanges are used instead of GENI. Periodic route reoptimizations are performed by means of an exact TSP algorithm. To help speed up computations, Taillard partitions the problem into several subproblems, each of which is solved independently on a parallel processor. In the case of planar problems, the decomposition process uses concentric rings carved into sectors centred at the depot. For non-planar problems a different decomposition method based on computation of shortest spanning arborescences is used. The boundaries of the subproblems are redefined dynamically.

Taillard's algorithm is one of the best available in terms of accuracy. It has identified twelve of the fourteen best known results on the CMT instances. Computation times needed to obtain best solutions are not reported and are difficult to establish with any degree of accuracy. The algorithm is, in one respect at least, simpler than Taburoute since it uses standard insertions, but managing the dynamic decomposition process as well as the parallel implementation adds to its complexity. One would expect this algorithm to handle additional side contraints reasonably well because of the combination of insertion and exchange moves used to define neighbour solutions. We believe, however, that resorting to simple insertions and a penalized objective function should offer more flexibility.

The adaptive memory procedure of Rochat and Taillard

The concept of adaptive memory, developed by Rochat and Taillard,[22] is probably one of the most powerful ideas put forward in the area of metaheuristics in recent years. An adaptive memory is a pool of good solutions produced by a heuristic, which is dynamically updated by adding to it new high quality elements and removing from it some of its least interesting elements. New elements are generated by recombining good solutions from the pool. The memory update process can therefore be viewed as a form of population search and as a generalization of genetic search. Rochat and Taillard[22] have developed an adaptive memory mechanism for the capacity and route duration constrained VRP and for the VRP with time windows, based on the earlier TS algorithms of Taillard[35] and Rochat and Semet.[61] Given a pool of good VRP solutions, new solutions are obtained by extracting high quality vehicle routes, where routes belonging to better solutions are given a higher probability of being selected. While extracting vehicle routes care is taken not to include those that contain already covered customers. Eventually this process will stop with a set of selected routes and some unrouted customers. A new solution is then reconstituted from these routes and unrouted customers using TS, and included in the pool if it is of sufficient quality. Two new best VRP solutions were identified through this process.

The success of an adaptive memory procedure is obviously linked to the capacity of the underlying search process to generate a pool of high quality solutions, which is certainly the case of Taillard's[35] TS algorithm. It may not work so well if an unsophisticated heuristic was employed to generate the individual solutions. The use of an adaptive memory certainly adds to computation time, as illustrated by some of the results obtained by Rochat and Taillard.[22] Coding an adaptive memory procedure requires a minimum of computing skills but is not overly complicated. The concept is highly flexible since it can be used in conjunction with other types of heuristics, not only TS, and it can easily be adapted to other contexts. For example, Bozkaya *et al*[62] report an application in the area of political districting.

The granular tabu search algorithm of Toth and Vigo

The idea behind granular tabu search (GTS)[63] is to remove from the graph unpromising arcs or edges that have only a small likelihood of belonging to an optimal solution. Toth and Vigo[63] eliminate all edges whose cost exceeds a *granularity threshold* $v = \beta\bar{c}$, where β is a *sparsification parameter*, and $\bar{c}$ is the average cost of an edge in a good solution generated with a fast heuristic. If β is chosen in the interval $[1.0, 2.0]$, then only 10–20% of the original edges tend to remain. The value of this parameter is dynamically updated throughout the search process. In their implementation, Toth and Vigo work on the restricted edge set $E(v) = \{(v_i, v_j) \in E : c_{ij} \leq v\} \cup I$, where I is a set of important edges such as those incident to the depot and those belonging to high quality solutions. The search mechanism implemented by Toth and Vigo uses Taburoute as a subroutine, but tends to produce better results faster. Because it works on a sparse graph, GTS can quickly perform an extensive search of the solution space. It also produces high quality solutions. The concept of granularity is relatively easy to implement once a good underlying search algorithm is available.

The unified tabu search algorithm of Cordeau et al

The unified tabu search algorithm (UTSA) was initially designed by Cordeau *et al* for the periodic VRP and the multi-depot VRP, and later slightly modified and extended to the single depot, multi-depot and periodic VRP with time windows by Cordeau *et al.*[64] The algorithm shares some features with Taburoute, namely a penalised objective function F' with self-adjusting coefficients to allow the exploration of intermediate infeasible solutions, and the use of continuous diversification. However, fixed length tabu durations are used, and only one initial solution is generated. As in Semet and Taillard,[65] the tabu mechanism operates on an attribute set $B(x)$ associated with solution x. More specifically, $B(x) = \{(i, k):$ vertex v_i is visited by vehicle k in solution $x\}$. Neighbour solutions are obtained by removing an attribute (i, k) from $B(x)$ and replacing it with (i, k'), where $k' \neq k$. The insertion of v_i into route k' is then performed so as to minimize the penalized objective function F', and attribute (i, k) is declared tabu for a set number of iterations.

The algorithm was highly successful on a variety of VRPs including the classical VRP with capacity and distance restrictions (see Table 3), the periodic and multi-depot VRPs,[5] the VRP with site dependencies,[4] and the single and multi-depot VRP with time windows.[64] On the classical VRP, it yields solution values within 0.69% of the best known. The average computing time is 13.75 min on a Sun Ultrasparc 10 workstation (440 MHz). On all other problems it often identified new best known solutions and has proved superior to several competing methods.

The UTSA ranks high on accuracy and performs rather well on speed, although it cannot compete on this dimension with very fast heuristics such as CW. Since all but one of its parameters are the same for all applications, this heuristic has in effect only one parameter. Moreover, since it operates according to a very simple mechanism, it is probably the simplest of all TS implementations for the VRP, but still it does not outperform CW in terms of simplicity. Finally, UTSA obtains excellent marks on flexibility. To our knowledge, this is the only VRP heuristic that can effectively

Table 3 Comparison of five metaheuristics for the VRP

			Taburoute[a]		*Taillard*[b]	*Adaptive memory*[b]	*Granular tabu search*[c]		*Unified tabu search algorithm*[d]		
Instance	*n*	*Type*	*Value*	*Minutes*[e]	*Value*	*Value*	*Value*	*Minutes*[f]	*Value*	*Minutes*[g]	*Best*
1	50	C	**524.61**[l]	*6.0*	**524.61**	**524.61**	**524.61**	*0.81*	**524.61**	*4.57*	**524.61**[i,k]
2	75	C	835.77	*53.8*	**835.26**	**835.26**	838.60	*2.21*	835.45	*7.27*	**835.26**[i]
3	100	C	829.45	*18.4*	**826.14**	**826.14**	828.56	*2.39*	829.44	*11.23*	**826.14**[i]
4	150	C	1036.16	*58.8*	**1028.42**	**1028.42**	1033.21	*4.51*	1038.44	*18.72*	**1028.42**[i]
5	199	C	1322.65	*90.9*	1298.79	**1291.45**	1318.25	*7.50*	1305.87	*28.10*	**1291.45**[j]
6	50	C,D	**555.43**	*13.5*	**555.43**	**555.43**	**555.43**	*0.86*	**555.43**	*4.61*	**555.43**[i]
7	75	C,D	913.23	*54.6*	**909.68**	**909.68**	920.72	*2.75*	**909.68**	*7.55*	**909.68**[i]
8	100	C,D	**865.94**	*25.6*	**865.94**	**865.94**	869.48	*2.90*	866.38	*11.17*	**865.94**[i]
9	150	C,D	1177.76	*71.0*	**1162.55**	**1162.55**	1173.12	*5.67*	1171.81	*19.17*	**1162.55**[i]
10	199	C,D	1418.51	*99.8*	1397.94	**1395.85**	1435.74	*9.11*	1415.40	*29.74*	**1395.85**[j]
11	120	C	1073.47	*22.2*	**1042.11**	**1042.11**	1042.87	*3.18*	1074.13	*14.15*	**1042.11**[i]
12	100	C	**819.56**	*16.0*	**819.56**	**819.56**	**819.56**	*1.10*	**819.56**	*10.99*	**819.56**[i]
13	120	C,D	1573.81	*59.2*	**1541.14**	**1541.14**	1545.51	*9.34*	1568.91	*14.53*	**1541.14**[i]
14	100	C,D	**866.37**	*65.7*	**866.37**	**866.37**	**866.37**	*1.41*	866.53	*10.65*	**866.37**[i]
Average deviation from best and time			0.86%	*46.8*	0.06%	0.00%	0.69%	*3.84*	0.69%	*13.75*	

[a]Standard algorithm with one set of parameters. Verifiable results. Computations performed by rounding costs five digits after the decimal point.[25]
[b]Verifiable results. Best of several runs. Computation times are not reported. Computations performed with floating point arithemic.[68]
[c]Toth and Vigo.[63] Computations performed by rounding costs three digits after the decimal point.[69]
[d]Solutions obtained by means of the Unified Tabu Search Algorithm of Cordeau *et al*,[5] without changing the parameters, and with 100 000 iterations. Computations performed with double-precision floating point arithmetic.
[e]Silicon Graphics workstation (36 MHz, 5.7 Mflops).
[f]Pentium 200 MHz PC.
[g]Sun Ultrasparc 10 (440 MHz).
[h]C: capacity restrictions, D: route duration restrictions.
[i]Taillard.[35]
[j]Rochat and Taillard.[22]
[k]Optimal solution value.[26]
[l]Bold numbers correspond to best known values.

Table 4 Assessment of some of the main VRP heuristics

Classical heuristics	*Accuracy*	*Speed*	*Simplicity*	*Flexibility*
Clarke and Wright (CW)[16]	Low	Very high	Very high	Low
Two-matching based methods[43]	High	Very low	Low	Low
Sweep[17]	Low	Medium-high	High	Low
1-Petal[47]	Low	High	Medium	Medium
2-Petal[30]	Medium	Medium	Medium	Medium
Fisher and Jaikumar (FJ)[18]	Difficult to assess	Medium	Low	Low
Location based FJ by Bramel and Simchi-Levi[51]	Medium	Low	Low	Low
Metaheuristics	*Accuracy*	*Speed*	*Simplicity*	*Flexibility*
Taburoute[25]	High	Medium	Medium	High
Taillard[35]	Very high	Low[b]	Medium-low	High
Adaptive memory[22,a]	Very high	Low[b]	Medium-low	High
Granular tabu search[63,a]	High	Medium	Medium	High
Unified tabu search algorithm[64]	High	Medium	Medium	High

[a]These are not VRP algorithms *per se*, but mechanisms that can be incorporated within another algorithm to improve its performance. Some of the ratings of these two features are therefore linked to those of the underlying algorithm. They relate to the results presented by Rochat and Taillard[22] and Toth and Vigo[63] for specific implementations.
[b]Taillard[35] and Rochat and Taillard[22] do not provide the time required to obtain their best known solution. They do, however, report the time required to obtain a solution value within 1% or 5% of the best known. These computation times can be rather high.

handle so many different variants with a unique set of algorithmic rules and a single user-controlled parameter.

We present in Table 3 comparative computational results for the various VRP metaheuristics just described. Care must be taken when comparing results since the Taillard and adaptive memory solution values are the best over several runs, whereas in the case of Taburoute, GTS and UTSA, all reported values correspond to a single run using standard parameter settings. In the case of Taburoute and UTSA, better solution values (sometimes corresponding to the best) were obtained by either changing parameter values (for Taburoute) or by executing several runs with the same parameters, starting from different randomly generated initial solutions (for UTSA).

Summary and conclusion

In this guide to VRP heuristics we have examined some of the well-known classical methods as well as some of the best available metaheuristics. Contrary to a long established custom of assessing heuristics against accuracy and speed only, we have added what we perceive as two criticial criteria: simplicity and flexibility. The results of our analysis are summarized in Table 4.

None of the classical heuristics fares very well on accuracy and flexibility. In this category, the celebrated Clarke and Wright heuristic has at least the distinct advantage of being very quick and simple to implement. This probably explains its ongoing popularity. However, in contexts where vehicle routes must be planned over a long horizon and large sums of money are at stake, it is worth investing time and resources in a method that performs a more extensive exploration of the search space.

Among the tabu search algorithms, there has been a tendency in recent years toward increased speed and simplicity. The adaptive memory procedure and the granularity principle are highly portable concepts that can be used within almost any search process. Among stand-alone heuristics, the unified tabu search algorithm scores very well on all dimensions.

Acknowledgements—This paper was partly written while the third author visited the Centre for Traffic and Transportation at the Technical University of Denmark. This research was partly funded by the Canada Research Chair in Distribution Management, by Canadian Natural Sciences and Engineering Research Council (NSERC) under grants 227837-00, OGP00338816, OGP0039682, and OGP0036662, and by the Fonds pour la Formation de Chercheurs et l'Aide à la Recherche (FCAR) under grant 2002-ER-73080. This support is gratefully acknowledged. The authors thank Edgar Alberto Cabral and François Guertin for their help with programming, and Oli BG Madsen who provided some useful references. Thanks are due to two referees for their valuable comments.

References

1 Dantzig GB and Ramser JH (1959). The truck dispatching problem. *Mngt Sci* **6**: 80–91.
2 Gendreau M, Laporte G, Musaraganyi C and Taillard ÉD (1999). A tabu search heuristic for the heterogeneous fleet vehicle routing problem. *Comput Opns Res* **26**: 1153–1173.
3 Mingozzi A, Giorgi S and Baldacci R (1999). An exact method for the vehicle routing problem with backhauls. *Transport Sci* **33**: 315–329.

4 Cordeau J-F and Laporte G (2001). A tabu search heuristic for the site dependent vehicle routing problem with time windows. *INFOR* **39**: 292–298.
5 Cordeau J-F, Gendreau M and Laporte G (1997). A tabu search heuristic for the periodic and multi-depot vehicle routing problems. *Networks* **30**: 105–119.
6 Dror M, Laporte G and Trudeau P (1994). Vehicle routing with split deliveries. *Discr Appl Math* **50**: 239–254.
7 Golden BL and Assad AA (eds) (1988). *Vehicle Routing: Methods and Studies*. North-Holland: Amsterdam.
8 Fisher ML (1995). Vehicle routing. In: Ball MO, Magnanti TL, Monma CL and Nemhauser GL (eds). *Network Routing, Handbooks in Operations Research and Management Science*, vol. 8. North-Holland: Amsterdam, pp 1–33.
9 Desrosiers J, Dumas Y, Solomon MM and Soumis F (1995). Time constrained routing and scheduling. In: Ball MO, Magnanti TL, Monma CL and Nemhauser GL (eds). *Network Routing: Handbooks in Operations Research and Management Science*, vol. 8. North-Holland: Amsterdam, pp 35–139.
10 Crainic TG and Laporte G (eds) (1998). *Fleet Management and Logistics*. Kluwer: Boston.
11 Toth P and Vigo D (eds) (2001). *The Vehicle Routing Problem. SIAM Monographs on Discrete Mathematics and Applications*. SIAM Publishing: Philadelphia, PA.
12 Laporte G and Semet F (2002). Classical heuristics for the capacitated VRP. In: Toth P and Vigo D (eds). *The Vehicle Routing Problem*. *SIAM Monographs on Discrete Mathematics and Applications.* SIAM Publishing: Philadelphia, PA, pp 109–128.
13 Gendreau M, Laporte G and Potvin J-Y (2002). Metaheuristics for the capacitated VRP. In: Toth P and Vigo D (eds). *The Vehicle Routing Problem*. *SIAM Monographs on Discrete Mathematics and Applications.* SIAM Publishing: Philadelphia, PA, pp 129–154.
14 Cordeau J-F, Desaulniers G, Desrosiers J, Solomon MM and Soumis F (2002). The VRP with time windows. In: Toth P and Vigo D (eds). *The Vehicle Routing Problem*. *SIAM Monographs on Discrete Mathematics and Applications.* SIAM Publishing: Philadelphia, PA, pp 157–193.
15 Toth P and Vigo D (1998). Exact solution of the vehicle routing problem. In: Crainic TG and Laporte G (eds). *Fleet Management and Logistics*. Kluwer: Boston, pp 1–31.
16 Clarke G and Wright JR (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Opns Res* **12**: 568–581.
17 Gillett BE and Miller LR (1974). A heuristic algorithm for the vehicle dispatch problem. *Opns Res* **22**: 340–349.
18 Fisher ML and Jaikumar R (1981). A generalized assignment heuristic for vehicle routing. *Networks* **11**: 109–124.
19 Kirkpatrick S, Gellatt CD Jr and Vecchi MP (1983). Optimization by simulated annealing. *Science* **220**: 671–680.
20 Glover F (1986). Future paths for integer programming and links to artificial intelligence. *Comput Opns Res* **13**: 533–549.
21 Holland JH (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press: Ann Arbor, MI.
22 Rochat Y and Taillard ÉD (1995). Probabilistic diversification and intensification in local search for vehicle routing. *J Heuristics* **1**: 147–167.
23 Christofides N, Mingozzi A and Toth P (1979). The vehicle routing problem. In: Christofides N, Mingozzi A, Toth P and Sandi C (eds). *Combinatorial Optimization*. Wiley, Chichester, pp 315–338.
24 Barr RS, Golden BL, Kelly JP, Resende MGC and Stewart WR Jr (1995). Designing and reporting on computational experiments with heuristic methods. *J Heuristics* **1**: 9–32.
25 Gendreau M, Hertz A and Laporte G (1994). A tabu search heuristic for the vehicle routing problem. *Mngt Sci* **40**: 1276–1290.
26 Hadjiconstantinou EA, Christofides N and Mingozzi A (1995). A new exact algorithm for the vehicle routing problem based on *q*-paths and *k*-shortest paths relaxations. In: Gendreau M and Laporte G (eds). *Freight Transportation*. Baltzer: Amsterdam, pp 21–43.
27 Mole RH (1983). The curse of unintended rounding error: a case from the vehicle scheduling literature. *J Opl Res Soc* **34**: 607–613.
28 Ichoua S, Gendreau M and Potvin J-Y (2000). Diversion issues in real-time vehicle dispatching. *Transport Sci* **34**: 426–438.
29 Gendreau M, Laporte G and Semet F (2001). A dynamic model and parallel tabu search heuristic for real-time ambulance relocation. *Parallel Comput* **27**: 1641–1653.
30 Renaud J, Boctor FF and Laporte G (1996). An improved petal heuristic for the vehicle routing problem. *J Opl Res Soc* **47**: 329–336.
31 Golden BL, Wasil EA, Kelly JP and Chao I-M (1998). Metaheuristics in vehicle routing: In: Crainic TG and Laporte G (eds). *Fleet Management and Logistics*. Kluwer: Boston, pp 33–56.
32 Rego C and Roucairol C (1996). A parallel tabu search algorithm using ejection chains for the vehicle routing problem. In: Osman IH and Kelly JP (eds). *Meta-Heuristics: Theory and Applications*. Kluwer: Boston, pp 661–675.
33 Rego C (1998). A subpath ejection method for the vehicle routing problem. *Mngt Sci* **44**: 1447–1459.
34 Lin S (1965). Computer solutions of the traveling salesman problem. *Bell Syst Tech J* **44**: 2245–2269.
35 Taillard ÉD (1993). Parallel iterative search methods for vehicle routing problems. *Networks* **23**: 661–673.
36 Solomon MM (1987). Algorithms for the vehicle routing and scheduling problem with time window constraints. *Opns Res* **35**: 254–265.
37 Gaskell TJ (1967). Bases for vehicle fleet scheduling. *Opl Res Q* **18**: 281–295.
38 Yellow P (1970). A computational modification to the savings method of vehicle scheduling. *Opl Res Q* **21**: 281–283.
39 Nelson MD, Nygard KE, Griffin JH and Shreve WE (1985). Implementation techniques for the vehicle routing problem. *Comput Opns Res* **12**: 273–283.
40 Paessens H (1988). The savings algorithm for the vehicle routing problem. *Eur J Opl Res* **34**: 336–344.
41 Desrochers M and Verhoog TW (1989). A matching based savings algorithm for the vehicle routing problem. Les Cahiers du GERAD G-89-04, École des Hautes Études Commerciales: Montreal.
42 Altinkemer K and Gavish B (1991). Parallel savings based heuristic for the delivery problem. *Opns Res* **39**: 456–469.
43 Wark P and Holt J (1994). A repeated matching heuristic for the vehicle routing problem. *J Opl Res Soc* **45**: 1156–1167.
44 Wren A (1971). *Computers in Transport Planning and Operation.* Ian Allan: London.
45 Wren A and Holliday A (1972). Computer scheduling of vehicles from one or more depots to a number of delivery points. *Opl Res Q* **23**: 333–344.
46 Foster BA and Ryan DM (1976). An integer programming approach to the vehicle scheduling problem. *Opl Res Q* **27**: 307–384.
47 Ryan DM, Hjorring C and Glover F (1993). Extension of the petal method for vehicle routing. *J Opl Res Soc* **44**: 289–296.
48 Desrosiers J, Soumis F and Desrochers M (1984). Routing with time windows by column generation. *Networks* **14**: 545–565.

49 Fisher ML, Greenfield AJ, Jaikumar R and Lester JT III (1982). A computerized vehicle routing application. *Interfaces* **12**(4): 42–52.

50 Fisher ML, Jaikumar R and Van Wassenhove LN (1986). A multiplier adjustment method for the generalized assignment problem. *Mngt Sci* **32**: 1095–1103.

51 Bramel JB and Simchi-Levi D (1995). A location based heuristic for general routing problems. *Opns Res* **43**: 649–660.

52 Osman IH (1993). Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annal Opns Res* **41**: 421–451.

53 Dueck G and Scheuer T (1990). Threshold accepting: a general purpose optimization algorithm. *J Comput Phys* **90**: 161–175.

54 Dueck G (1993). New optimization heuristic: the great deluge algorithm and the record-to-record travel. *J Comput Phys* **104**: 86–92.

55 Van Breedam A (1996). An analysis of the effect of local improvement operators in genetic algorithms and simulated annealing for the vehicle routing problem. RUCA Working Paper 96/14, University of Antwerp: Belgium.

56 Bullnheimer B, Hartl RF and Strauss C (1999). An improved ant system algorithm for the vehicle routing problem. *Annal Opns Res* **89**: 561–581.

57 Ghaziri H (1996). Supervision in the self-organizing feature map: application to the vehicle routing problem. In: Osman IH and Kelly JP (eds). *Meta-Heuristics: Theory and Applications*. Kluwer: Boston, pp 651–660.

58 Willard JAG (1989). Vehicle routing using r-optimal tabu search, MSc dissertation, The Management School, Imperial College: London.

59 Gendreau M, Hertz A and Laporte G (1992). New insertion and postoptimization procedures for the traveling salesman problem. *Opns Res* **40**: 1086–1094.

60 Taillard ÉD (1991). Robust taboo search for the quadratic assignment problem. *Parallel Comput* **17**: 433–445.

61 Rochat Y and Semet F (1994). A tabu search approach for delivering pet food and flour in Switzerland. *J Opl Res Soc* **45**: 1233–1246.

62 Bozkaya B, Erkut E and Laporte G (2002). A tabu search heuristic and adaptive memory procedure for political districting. *Eur J Opl Res*, to be published.

63 Toth P and Vigo D (1998). The granular tabu search (and its application to the vehicle routing problem). *Technical report* OR/98/9, DEIS, Università di Bologna: Italy.

64 Cordeau J-F, Laporte G and Mercier A (2001). A unified tabu search heuristic for vehicle routing problems with time windows. *J Opl Res Soc* **52**: 928–936.

65 Semet F and Taillard ÉD (1993). Solving real-life vehicle routing problems efficiently using tabu search. *Annal Opns Res* **41**: 469–481.

66 Renaud J (2001). Private communication.

67 Simchi-Levi D (2001). Private communication.

68 Taillard ÉD (2001). Private communication.

69 Toth P (2001). Private communication.

Received June 2001;
accepted November 2001 after one revision